Scripted Lighting System: SLS2

Documentation for the Scripted Lighting System Version 2
for Neverwinter Nights 2.

Last Updated: **23-Sep-08**
For the latest version visit http://www.slayweb.com

A demonstration module showing the features explained here is available from my website here: http://www.slayweb.com/downloads/sls2-demo.zip

Contents

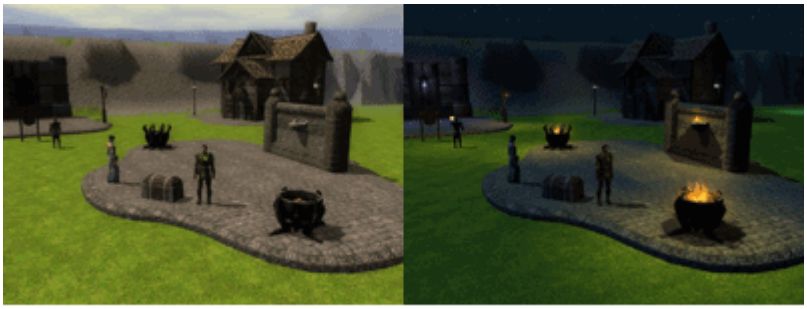
Introduction

The Scripted Lighting System Version 2 (SLS2) is a set of scripts, blueprints and prefabs for Neverwinter Nights 2 module builders. The system allows builders to setup complex lighting effects without the need for extensive scripting knowledge. All the main setup is done though variables.

The original SLS was a basic system which let builders set their lights to turn on and off at specified times of day allowing you to have an exterior area where the lights turned off during the day.

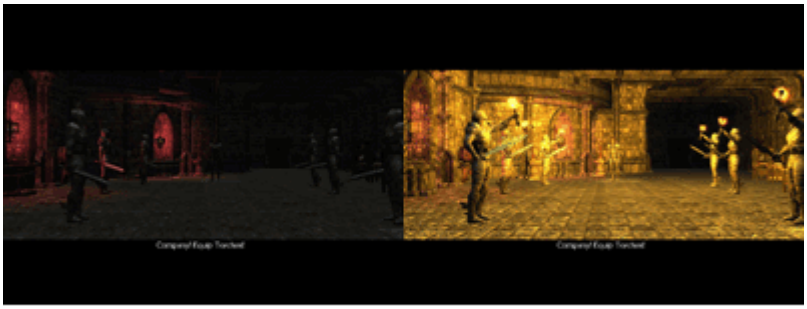
I saw room for improvement and started working on SLS2. This ended up being a complete rewrite with SLS2 becoming a complete lighting control and event system.

Of course SLS2 still allows you to set automatic day/night schedules for your lights but there are now many other ways of controlling what your lights do. Triggers, conversations and scripts can all be used to modify lights in different ways.
Here are some examples:



SET DAY/NIGHT SCHEDULES FOR YOUR LIGHTS

DIFFERENT LIGHTS AT DIFFERENT TIMES, INTERIOR DAY/NIGHT SETTINGS



CONTROL LIGHTS DURING CUTSCENES

Installation

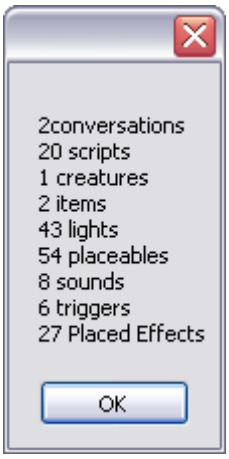
Inside the downloaded Zip file along with this documentation is a file called "sls2.erf". You need to import this file into your NWN2 module to use SLS2.

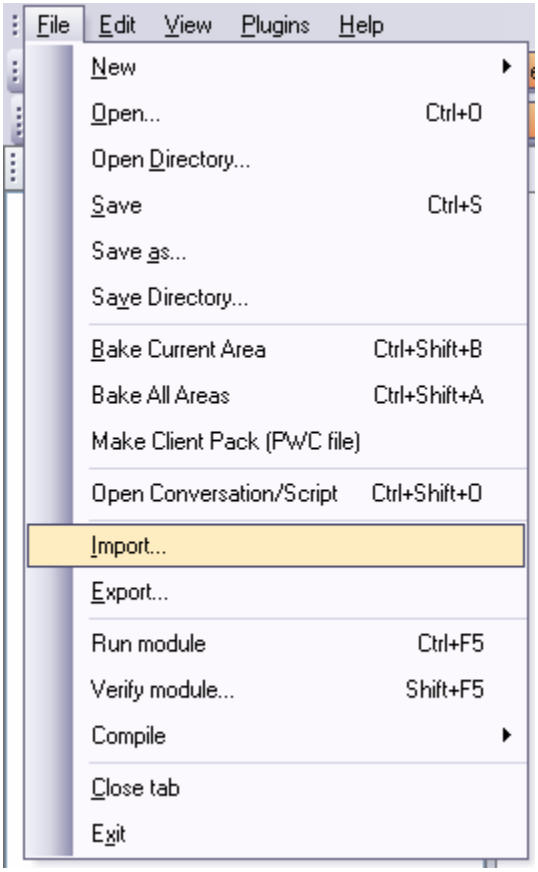
Open your module in the toolset.

Select "Import…" from the File menu.

Find the file "sls2.erf" on your computer, select it then click "Ok".

You will see a message when the import is complete, detailing what has been imported.





Also 109 lighting prefabs will have been installed which the toolset does not mention. You will need to restart the toolset before the prefabs will be available.

Once the toolset has been restarted the SLS2 installation will be complete and ready to use.

Upgrading from SLS1
SLS2 is a complete rewrite and therefore separate from SLS1. They can actually both be used at the same time. This allows you to take your time if you need to upgrade an existing module as you can go through area by area without breaking the module. Notes on upgrading your existing individual Light Fittings can be found later in the documentation.

Basic concepts

The principle concept behind SLS2 is called a Light Fitting. A Light Fitting is a group of game objects which together are used to create lighting effects.

There are four types of objects which are used to create a Light Fitting:

PLACEABLE

Placeables are the core object in a Light Fitting; every Light Fitting has to have one. However the placeable does not have to be a lighting related one, it could just as easily be a crate. The placeable contains the majority of the setting variables for the Light Fitting.

VISUAL EFFECT

There are many visual effects available in the game nearly all of which represent some kind of light effect. The most commonly used for area lighting are of course the burning/fire effects, used along with torches, braziers & bonfires etc.
Visual effects are essentially just an image and don't produce any actual light.

LIGHT

Light objects provide the actual light from a Light Fitting. They can have colour, brightness and flicker setting changed for a wide variety of lighting types.
Nearly all Light Fittings will have a light object but there are occasions where they're not needed.

SOUND

Sometimes a Light Fitting will require a related sound effect to play while the light is on, such as the crackle of a fire burning or some sort of magical hum.
Sounds are not included on the provided prefabs for two reasons, one being that it's not possible and two being that sounds are usually better spread generally around rather than being specific to an object.

Once a combination of these four objects is put together you have a Light Fitting. The basic setup procedure for a Light Fitting (explained in detail later) is as follows:
First place the objects in your area. Next give the visual effects, lights & sounds unique tags. Then enter those tags in the placeables variable list. Finally add on/off times to the placeables variables and add the user defined event script.

Light setup

Before you start to add lights to your area you must first setup two scripts in the area properties.
Add `sls2_onenter` to "On Client Enter Script" and add `sls2_heartbeat` to "On Heartbeat Script".

Scripts	
On Client Enter Script	sls2_onenter
On Enter Script	
On Exit Script	
On Heartbeat Script	sls2_heartbeat
On User Defined Event Script	
Variables	

If your area is already using scripts for these events then you will have to modify those scripts instead.

In this case you need to add a single line of code to the end of your existing scripts.

Add the following to your "On Client Enter Script":
```
ExecuteScript("sls2_onenter",OBJECT_SELF);
```

Add the following to your "On Heartbeat":
```
ExecuteScript("sls2_heartbeat",OBJECT_SELF);
```

Once that's done you're ready to start adding lights to your area. There are four major ways to do this, detailed below.

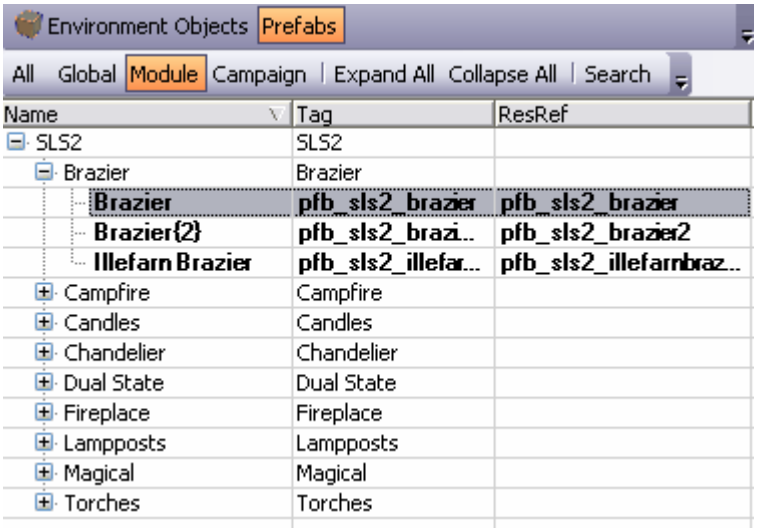
About Heartbeats

Some people are often concerned with the performance of heartbeat scripts especially in Persistent Worlds. The SLS2 heartbeat is designed to do as little as possible. However if you do decide that you'd rather not run the heartbeat you can still use many of SLS2's features. The SLS2 heartbeat is only used to control Light Fittings set to be on/off at different times of day. Most of the SLS2 features such as controlling lights though scripts, conversations and triggers do not require the heartbeat.

Light setup - prefabs

SLS2 comes with many prefabs of Light Fittings which are nearly complete and require very little setup before they can be used.

These are all available in the Prefabs section of your blueprints palette in the SLS2 category.



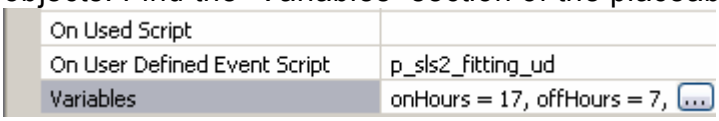
Select a prefab from the list and place it into your area.

Once you have the prefab in the desired position, right click it and select “Ungroup”.

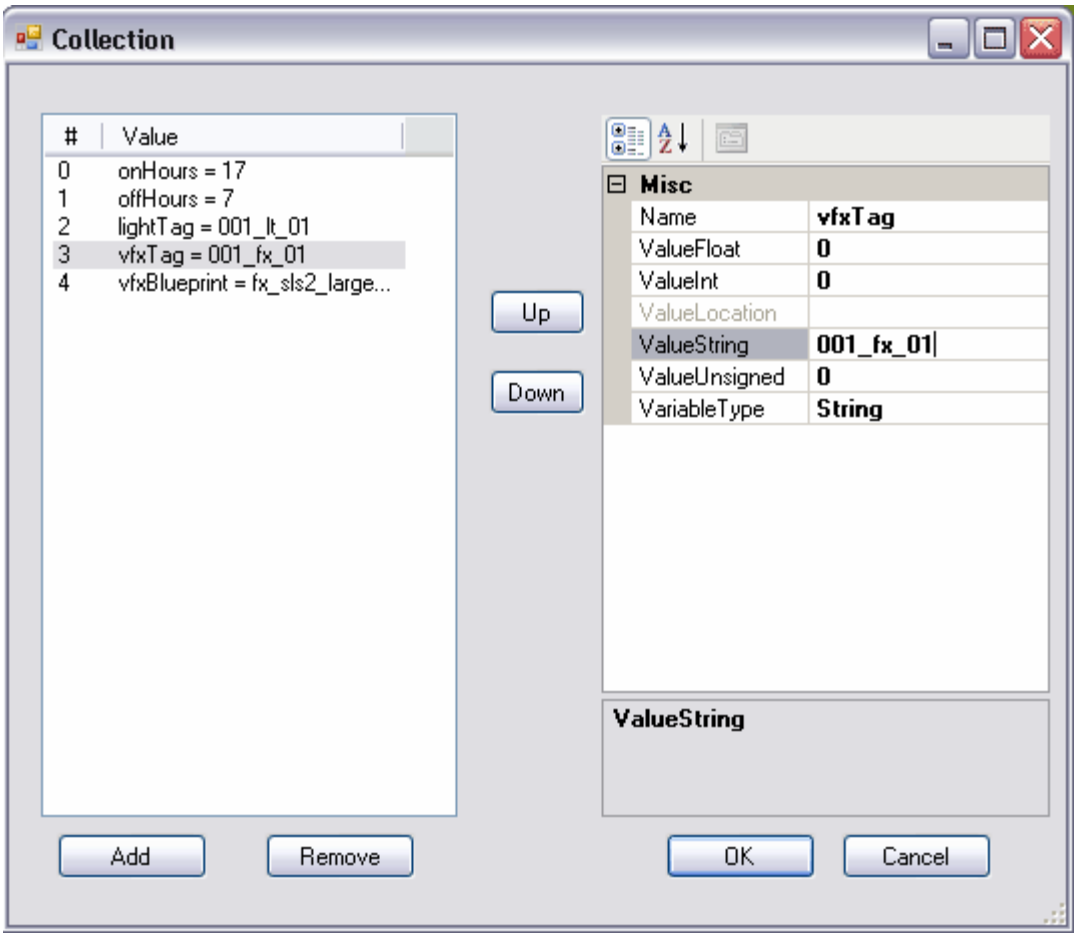
The previously grouped objects will now all be editable separately.

The light and visual effect objects must now be given a unique tag.
The naming scheme you use for your tags is up to you. I use a sequential list grouped by area and object type, `areanum_objecttype_fittingnum`.
In this case the tags would be `001_lt_01` for the light object and `001_fx_01` for the visual effect object.

Now the placeable needs to know the tags you have chosen for its related objects. Find the “Variables” section of the placeable’s properties.



You will see that the SLS2 scripts and default variables are already set on the provided prefabs. Click the “…” button to edit the variables.



Add the tag you chose for the light object to the “lightTag” variable and add the tag you chose for the visual effect object to the “vfxTag” variable.
Both values are strings. Click “OK” once you’re done.

The setup for this light is now complete. The light will turn on at night and off during the day, it will also respond to light events from other sources.

Regardless of how you want the lights to work in game the Light Fittings in the toolset are always built in the on state.

Light Setup - Parts

An alternative to using the SLS2 prefabs is to make your own Light Fittings using the provided blueprints allowing you to put your own combinations together quickly.

Select one of the SLS2 placeables from the SLS2 category in the placeables section of the blueprint palette and then add it to your area.

Name	Tag
SLS2	SLS2
Dual State	Dual State
Other	Other
Standard	Standard
Brazier	Brazier
Campfire	Campfire
Bonfire	PLC_NT_BON...
Camp Fire {01}	PLC_MR_CAM...
Camp Fire {04}	PLC_MR_CAM...
Cooking Spit...	PLC_ML_SPIT01
Cooking Spit...	PLC_ML_SPIT02
Candles	Candles
Chandelier	Chandelier
Fireplace	Fireplace
Lampposts	Lampposts
Magical	Magical
Torches	Torches
Useable	Useable

Select one of the SLS2 visual effects from the placed effects section of blueprint palette. You'll find them in the SLS2 category. Add one to your area and move it into position.

Name	ResRef
SLS2	
Fire	
Bonfire	fx_sls2_bonfire
Candle Flame	fx_sls2_candle
Fire Cone	sls2_fx_firecon
Large Brazier Fire	fx_sls2_largefi
Large Torch (Blue)	fx_sls2_largeto
Small Fire	fx_sls2_smallfi
Torch Fire	fx_sls2_torchfi
Torch Fire (Aqua)	fx_sls2_torchfi
Torch Fire (Blue)	fx_sls2_torchfi
Torch Fire (Green)	fx_sls2_torchfi
Torch Fire (Orange)	fx_sls2_torchfi
Torch Fire (Pink)	fx_sls2_torchfi
Torch Fire (Purple)	fx_sls2_torchfi
Torch Fire (Red)	fx_sls2_torchfi
Torch Fire (White)	fx_sls2_torchfi
Torch Fire (Yellow)	fx_sls2_torchfi
Glow	

Choose one of the SLS2 light blueprints from lights section of the palette. Add it to your area and then move it into position.

Name	ResRef
SLS2	
Brazier	
Fire	
3 Candles	lt_sls2_candle3
5 Candles	lt_sls2_candle5
Bonfire	lt_sls2_bonfire
Campfire	lt_sls2_campfire
Chandelier 1	lt_sls2_chandel...
Chandelier 2	lt_sls2_chandel...
Fireplace	lt_sls2_fireplace
Large Torch (Bl...	lt_sls2_largetor...
Single Candle	lt_sls2_candle
Tiki Torch	lt_sls2_tiki
Tiki Torch (Aqua)	lt_sls2_tiki_aqua
Tiki Torch (Blue)	lt_sls2_tiki_blue
Tiki Torch (Gre...	lt_sls2_tiki_green
Tiki Torch (Pink)	lt_sls2_tiki_pink
Tiki Torch (Pur...	lt_sls2_tiki_pur...

Finally select a sound to add from the blueprint palette. These are in the SLS2 category of the sounds section.

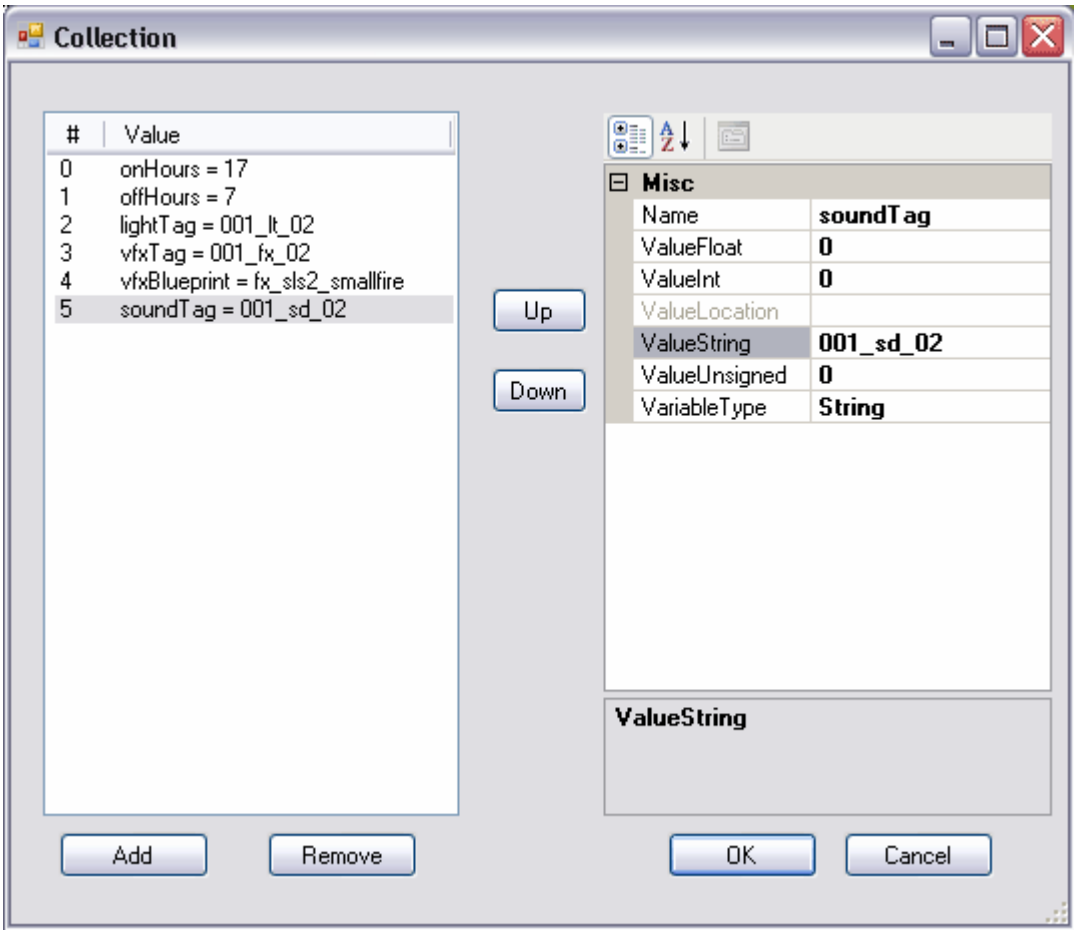
Name	ResRef
SLS2	
Fire	
Brazier Fire	sd_sls2_brazie...
Fire Bowl	sd_sls2_firebowl
Fire Large 2	sd_sls2_firelar...
Fire Medium 1	sd_sls2_fireme...
Fire Smolder	sd_sls2_firesm...
Firepit	sd_sls2_campfire
Fireplace	sd_sls2_fireplace
Torch Fire Small	sd_sls2_torchfire

The light, effect and sound objects must now be given unique tags.
The naming scheme you use for the tags is up to you. I use a sequential list grouped by area and object type, `areanum_objecttype_fittingnum`.
In this case the tags would be `001_lt_02` for the light object, `001_fx_02` for the visual effect object and `001_sd_02` for the sound object.

Now the placeable needs to know the tags you have chosen for its related objects. Find the “Variables” section of the placeable’s properties.

On Used Script	
On User Defined Event Script	p_sls2_fitting_ud
Variables	onHours = 17, offHours = 7, ...

You will see that the necessary scripts and default variables are already set on the provided SLS2 blueprints. Click the "…" button to edit the variables.



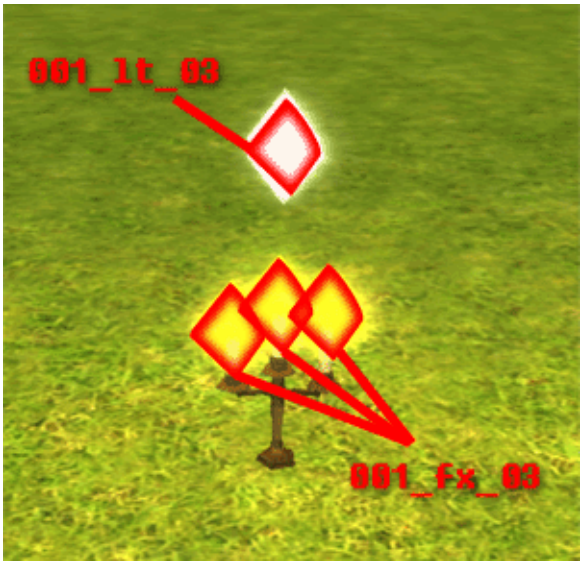
Add the tags you have chosen for the light and visual effects into the provided blank variables "lightTag" and "vfxTag".
Space for sound information is not included by default. If you have decided to add a sound you will need to add a new variable of string type called "soundTag" and then add your chosen tag to it.
Also check the "vfxBlueprint" variable, if the default is not the same as the one you have selected you will need to update this variable to the resref of the effect you are using.

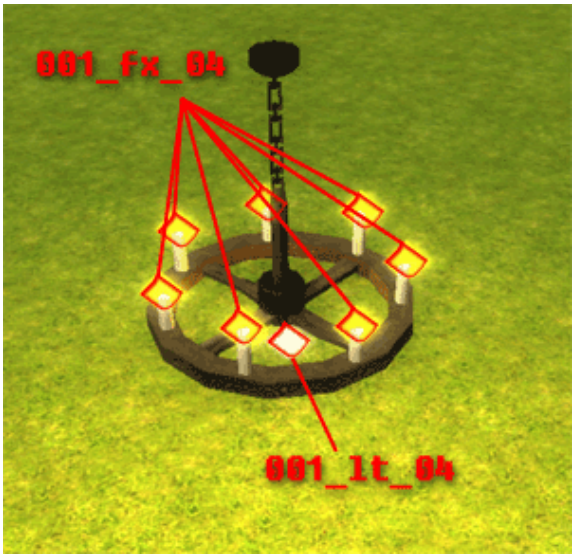
The setup for this light is now complete. The light will turn on at night and off during the day, it will also respond to light events from other sources.

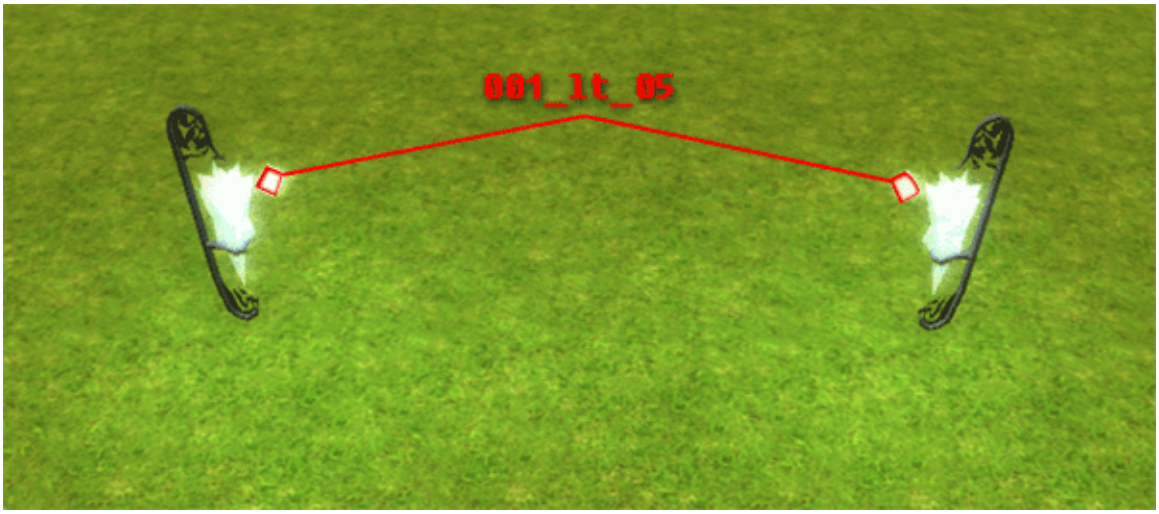
When creating Light Fittings all objects except the placeable are optional, though to actually observe the effect you'll need at least one of them. Remove the variables related to the object you're not using from the placeable so the scripts won't try to process them.

Light Fittings can have more than one effect or light object if necessary. To do this all the objects of the same type, related to the same fitting, have the same tag.

Some examples:







Light Setup - Upgrading from SLS1

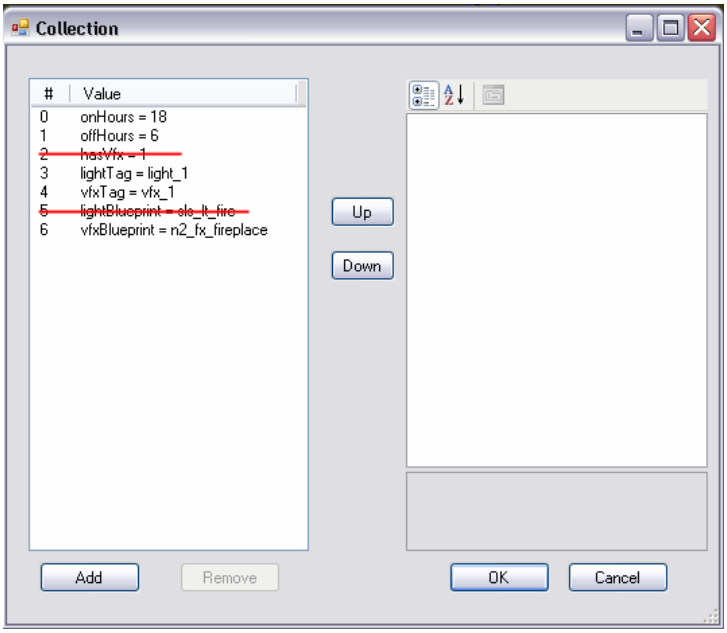
SLS2 has changed a lot at its core compared to the original SLS. It has therefore proved impossible to provide a direct upgrade for modules currently using SLS1. This is basically a replacement rather than an upgrade.

Many of the settings are the same but the script names have changed, so I'm afraid every Light Fitting will need to be updated. However because SLS2 is a separate code base both versions can run simultaneously across a module, each controlling different areas. This allows you to make the conversion in stages.
(Note: running both versions together has not been extensively tested and is not advised for a final production module)

First you need to update the scripts in the area properties.
Remove `sls_onenter` from "On Enter Script"
Add `sls2_onenter` to "On Client Enter Script"
Change `sls_heartbeat` to `sls2_heartbeat`

For each Light Fitting placeable, access its properties and update "On User Defined Event" to `p_sls2_fitting_ud`.
The Light Fitting placeables are no longer required to have the specific tag of "lightfitting". This can be changed now if desired allowing specific Light Fittings to be accessed by scripts and triggers etc.
Finally check the placeables variables; these should work correctly as they are currently set. However if you want to keep your module tidy it should be noted that the variables "hasVfx" and "lightBlueprint" are no longer required.

That's all that needs to be done to old SLS1 lights. Though I admit it will be a tedious task if you have a lot of lights, sorry about that.



Light Setup - Non-SLS Placeables

You can build a Light Fitting with any of the game objects, not just the ones provided in the SLS2 package.
The process is mostly the same as using SLS2 parts. You add your placeable along with a combination of lights, effects & sounds, then give them unique tags. You now need to add the tags to the variables on the placeable, but when you're not using one from the SLS2 package it won't have the defaults already available. You'll have to add all the variables yourself. Open up the variables screen for the placeable and add the following:
onHours – *Integer*: **17**
offHours – *Integer*: **7**
lightTag – *String*: **the tag of the related lights**
vfxTag – *String*: **the tag of the related visual effects**
soundTag – *String*: **the tag of the related sounds**
vfxBlueprint – *String*: **the resref of the visual effect being used**

Finally add the script `p_sls2_fitting_ud` to the placeables "On User Defined Event Script"

This is all you need to get your custom placeables working with SLS2.

See the "Variables" section for more information about all the available variable options and when they are required.

Dual State Lights

Some of the lighting placeables are available in two versions, one for on and another for off. With SLS2 you can set a Light Fitting to use both placeables and spawn the right one for the current light state.
To do this we need to introduce two new variables, "onBlueprint" and "offBlueprint".

Adding these variables to a Light Fitting placeable will tell the Light Fitting to spawn the relevant placeable depending on whether the light is on or off.

Both variables should be a string and should contain the resref of the placeable you want to spawn at that time.

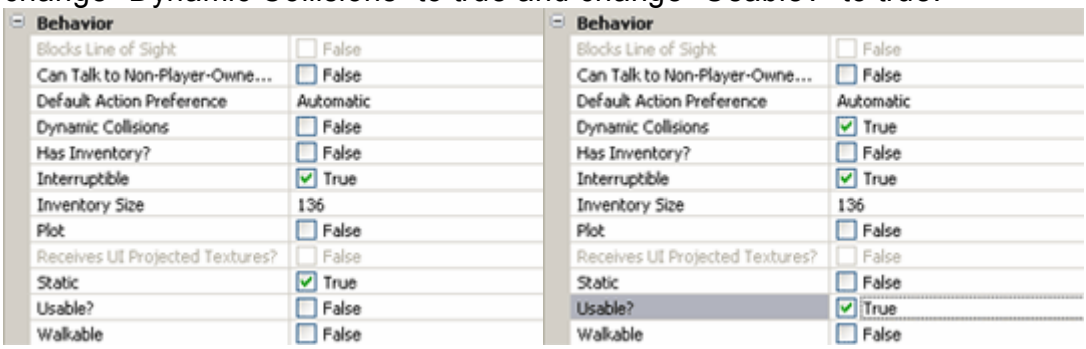
There are prefabs and blueprints for all the available dual state lights in the retail game with the correct variables already set.

Useable Lights

Using SLS2 it is also possible to make a Light Fitting useable. When a PC uses the placeable in game the light will toggle on or off.

Doing this requires you to change some of the placeable settings and add an extra script.

Go to the “Behavior” section of placeable settings, change “Static” to false, change “Dynamic Collisions” to true and change “Usable?” to true.



To complete the setup add the script `p_sls2_fitting_us` to the placeables “On Used Script”. The Light Fitting will then be useable in game.

A useable dual state light is a little more complicated as it requires a special useable version of the “offBlueprint” and “onBlueprint” for spawning. See the included ghostly light example.

Light chains

A light chain is a group of Light Fittings which are set to light up in a sequence.

To create a chain set the tags of all the Light Fittings to be in the chain as you would a set of waypoints. E.g. light1, light2, light3, ..., lightN
The numbering will define the order in which they light up.

Each Light Fitting in the chain can have the variable “chainDelay” added. This is a float variable which controls how long (in seconds) the Light Fitting should stay on during its turn in the chain. If this variable is omitted the default timing of 0.5 seconds will be used.

Light chains are started and stopped by triggers. In the triggers section of your blueprints palette you’ll find two SLS2 triggers for chains, one to start chains and the other to stop them.

Name	Tag	ResRef
SLS2	SLS2	
Chain Start Trigger	SLS2ChainStar...	sls2_chainstart
Chain Stop Trigger	SLS2ChainSto...	sls2_chainstop
Off Trigger	SLS2OffTrigger	sls2_offtrigger
On Trigger	SLS2OnTrigger	sls2_ontrigger0
One Shot Trigger	SLS2OneShotT...	sls2_oneshottri...
Pressure Trigger	SLS2Pressure...	sls2_pressuretr...

Add a “Chain Start Trigger” to your area at the point where you want the PC to initiate the light chain. The trigger has three/four variables that need to be set for the chain to start correctly.
chainTag - *String*: **the tag you gave to the Light Fittings, without the number**
chainLength - *Integer*: **the number of Light Fittings in the chain**
chainType - *String*: **how the chain acts, see below**
chainDir - *Integer*: **0/1, the direction of the chain (optional, uses 0 if omitted)**

The values that you choose for “chainType” and “chainDir” define how the chain displays.

“chainType” can have the value of `loop` or `bump` which decides how the chain reacts at its ends. A chain set to `loop` will go back to position 1 after it gets to the end. While a chain set to `bump` will return along the reverse path when it gets to the end.

“chainDir” can have the value or 0 or 1 (0 is assumed if missing). Zero represents up so a chain with this setting will follow your numbers in ascending order while one sets it to the reverse.

To stop a chain you need to add a “Chain Stop Trigger”. This trigger needs the “chainTag” variable set so it can stop the correct chain.

Creatures / npcs

It is also possible to use creatures and npcs as Light Fittings. They do not require associated light and visual effect objects, but will instead equip a torch when their light state is set to be on.

To set a creature to be controlled by SLS2 you need to do two things. The first is to set the “On User Defined Event Script” for the creature to be `b_sls2_npc_ud`. Second you need to add the variables “onHours” and “offHours” to the creature.
onHours – *Integer*: **17**
offHours – *Integer*: **7**

With this complete the creature will equip a torch when set to be “on”.

Advanced concepts

The majority of the SLS2 functions work by using the user defined events system. Light events are sent from a script or object and then the Light Fitting itself processes the event and decides what to do.

There are 10 different events defined by SLS2. Any object with the script `p_sls2_fitting_ud` or creature with the script `b_sls2_npc_ud` will process these events.

All the constants for these events can be found in the global include script: `ginc_sls2`

INIT EVENT
The init event is sent by the "On Client Enter Script" for the current area. It only runs once when the first PC visits an area and tells all the Light Fittings to initialize themselves.

TOGGLE EVENT
When a Light Fitting receives a toggle event it will switch to its opposite state.
A useable light sends a toggle event to itself.

TURN ON EVENT
This event tells a light to turn on. If the light is off it will turn on. If the light is on it will do nothing and stay on.

TURN OFF EVENT
This event tells a light to turn off. If the light is on it will turn off. If the light is off it will do nothing and stay off.

TURN ON PERMANENT EVENT
The permanent version of the turn on event tells the light to turn on in the same way as the basic turn on event. In addition to that it sets the light to ignore its "onHours" and "offHours" so it will no longer respond to "Update" events and will stay on while disregarding its schedule.

TURN OFF PERMANENT EVENT
The permanent version of the turn off event tells the light to turn off in the same way as the basic turn off event. In addition to that it sets the light to ignore its "onHours" and "offHours" so it will no longer respond to "Update" events and will stay off while disregarding its schedule.

UPDATE EVENT
The update event instructs the Light Fitting to check its "onHours" and "offHours" variables and decide if it should be on or off. The Light Fitting will then set its light state accordingly.
This is how scheduled lights function. The script `sls2_heartbeat` sends this event to lights once every game hour.

RESET EVENT
Sending a reset event causes the Light Fitting to reanalyze its variables and set itself back to its default state.

ONE SHOT EVENT
A one shot event will cause a Light Fitting to turn on, wait for a specified delay and then turn off again. The delay is specified in the float variable “oneShotDelay” on the Light Fitting placeable. This event can be useful to add extra lighting to a non-looping visual effect.

CHAIN EVENT
A chain event is not meant to be sent manually. Chain events are sent automatically by lights during a chain sequence. Light chains have to be started by a trigger or script call.

With the default SLS2 scripts in place some of these events are sent automatically. Init events are sent by script `sls2_onenter` when the first PC enters an area. Updates events are sent by script `sls2_heartbeat` once every game hour, this controls the scheduled lights. Also toggle events are sent by lights which are set to be useable and chain events are sent by lights in a currently running chain.

These events can also be sent manually by triggers, conversations and scripts.

Triggers
Included in the SLS2 package are six new triggers for use in controlling lights. The “Chain Start” and “Chain Stop” triggers have already been covered in the Chain Lights section.
The other four triggers are: On, Off, Pressure and One Shot.

Name	Tag	ResRef
SLS2	SLS2	
Chain Start Trigger	SLS2ChainStar...	sls2_chainstart
Chain Stop Trigger	SLS2ChainSto...	sls2_chainstop
Off Trigger	SLS2OffTrigger	sls2_offtrigger
On Trigger	SLS2OnTrigger	sls2_ontrigger0
One Shot Trigger	SLS2OneShotT...	sls2_oneshottri...
Pressure Trigger	SLS2Pressure...	sls2_pressuretr...

ON TRIGGER
An on trigger will signal one or more Light Fittings to turn on when a PC activates it.
To use an on trigger give all the Light Fittings you want to activate the same tag. Then enter that tag into the “fittingTag” variable located on the trigger. The on trigger sends a “Permanent On” event so the affected fittings will stay on until they receive another event.

OFF TRIGGER
An off trigger works in the same way as an on trigger, just sending an off event instead.

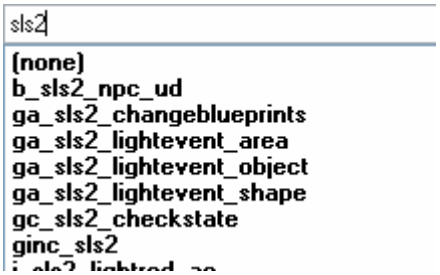
PRESSURE TRIGGER
Pressure triggers combines an on and off trigger together only activating the lights while the PC is on the trigger. It turns the lights on at “On Enter” and off again at “On Exit”.

ONE SHOT TRIGGER
A one shot trigger sends a one shot event to Light Fittings with its “fittingTag”. A Light Fitting requires a “oneShotDelay” float variable to respond to one shot events. The delay variable defines how long the Light Fitting stays on.

Conversations

There are five scripts included with SLS2 which are for attaching to conversation nodes. Four are global actions while the other is a global conditional.

With the relevant conversation node selected go to the actions or conditions tab to and click “Add” to add the scripts.
Search for `sls2` to find the scripts quickly.



Actions begin with `ga_` and conditions begin `gc_`.

The scripts and their parameters are detailed below.

ga_sls2_lightevent_area
nEvent (Integer)
sAreaTag (String)
This action sends a light event to all objects in an area. Parameter nEvent defines the SLS2 event number to be sent. Parameter sAreaTag defines the tag of the area the event should be sent to. If sAreaTag is left blank then the location of the current PC speaker will be used.

ga_sls2_lightevent_shape
nEvent (Integer)
fRange (Float)
This action sends a light event to all objects within a sphere shape with the current PC speaker at the centre. Parameter nEvent defines the SLS2 event number to be sent. Parameter fRange defines the size of the sphere.

ga_sls2_lightevent_object
nEvent (Integer)
sObjectTag (String)
This action sends a light event to all objects with the tag specified by parameter sObjectTag. Parameter nEvent defines the SLS2 event number to be sent.

ga_sls2_changeblueprints
sTag (String)
sNewLightBP (String)
sNewVfxBP (String)
bDelay (Integer)
This action uses the change blueprints feature of SLS2 to alter the blueprint information stored on a Light Fitting allowing you change the light and effect objects in use. This can be used to change colours and types quickly. Parameter sTag represents the tag of the Light Fittings which are to be changed. Parameter sNewLightBP is a string defining the resref of the new light to be used. Parameter sNewVfxBP is a string containing the resref of the new visual effect to be used. If parameters sNewLightBP or sNewVfxBP are left blank then that type of object will not be changed. Parameter bDelay is a boolean integer (0 or 1). When

bDelay is set to 0 all Light Fittings will be updated at the same time, when set to 1 the Light Fittings will be updated with a delay between each update. The delay is defined by the constant `SLS2_EVENT_DELAY` located in `ginc_sls2`.

gc_sls2_checkstate
nState (Integer)
sFitting (String)
This conditional checks the state of the Light Fitting with the tag specified in parameter sFitting. It returns "true" it the Light Fittings current state matches the state chosen in parameter nState, otherwise it returns "false".

Scripting

You can also control the lighting in your module directly from your own scripts. All the SLS2 functions are available in a single include file ready for your use.

Add `#include "ginc_sls2"` to the header of your script to access the SLS2 functions.

Not all of the functions available in the system are intended for direct use. Following is a list of the main functions designed for scripting access:

```
// Loops all objects in an area and sends the specified
// light event to them
// nEvent: the SLS2 light event constant to send
// oArea: the area to send the event to
// bDelay: TRUE - Send the events with a delay in between
//         FALSE - Send the events all at once
void SLS2AreaLightEvent(int nEvent, object oArea=OBJECT_INVALID, int
bDelay=TRUE)

// Send a light event to all objects with the specified tag
// nEvent: the SLS2 light event to send
// sObjectTag: the tag of the objects which want the event
void SLS2ObjectLightEvent(int nEvent, string sObjectTag)

// sends a light event to all object within a sphere shape
// nEvent: the sls2 light event to send
// fRange: the sphere radius
// lLocation: the location of the centre of the sphere
void SLS2ShapeLightEvent(int nEvent, float fRange, location lLocation)

// change the light/vfx blueprints for the specified objects
// sTag: the tag of the objects to change
//       will attempt to change OBJECT_SELF if blank
// sNewLightBP: new light blueprint, no change if blank
// sNewVfxBP: new vfx blueprint, no change if blank
// bDelay: TRUE - Send the updates with a delay in between
//         FALSE - Send the updates all at once
void SLS2SetBlueprints(string sTag="",string sNewLightBP="",string
sNewVfxBP="",int bDelay=0)
```

```
// print out the current time, properly zero padded
void SLS2SpeakTime()

// start a set of chain lights running
// sTag: the tag of the lights in the chain
// nLength: the number of lights in the chain
// sType: the type of the chain (loop/bump)
// nDir: the direction of the chain (0/1)
void SLS2StartChain(string sTag,int nLength,string sType,int nDir)

// stop a set of chain lights from running
// sTag: the tag of the lights in the chain
void SLS2StopChain(string sTag)
```

Also of interest are the default SLS2 constants for light states and light events.

```
// Light States
const int SLS2_LIGHTSTATE_ON         = 220000;
const int SLS2_LIGHTSTATE_OFF        = 220001;
const int SLS2_LIGHTSTATE_PERMAON    = 220002;
const int SLS2_LIGHTSTATE_PERMAOFF   = 220003;
// User Defined Events
const int SLS2_EVENT_INIT            = 221000;
const int SLS2_EVENT_TOGGLE          = 221001;
const int SLS2_EVENT_TURNON          = 221002;
const int SLS2_EVENT_TURNOFF         = 221003;
const int SLS2_EVENT_TURNONPERM      = 221004;
const int SLS2_EVENT_TURNOFFPERM     = 221005;
const int SLS2_EVENT_UPDATE          = 221006;
const int SLS2_EVENT_RESET           = 221007;
const int SLS2_EVENT_ONESHOT         = 221008;
const int SLS2_EVENT_CHAIN           = 221009;
// Other
const float SLS2_EVENT_DELAY         = 0.1f;
```

Variable reference

This is a summary of all setting variables used by SLS2 placeables along with a short description of what they do and when they are required.

Variable Name	Variable Type	Description
onHours	Integer	The time at which the Light Fitting should switch on. If you do not want the light to be scheduled you can omit this variable and the light will default to being permanently on.
offHours	Integer	The time at which the Light Fitting should switch off. If you do not want the light to be scheduled you can omit this variable and the light will default to being permanently on.
lightTag	String	The tag of all the light objects related to this Light Fitting. Not required if the Light Fitting does not have any light objects.
vfxTag	String	The tag of all the visual effect objects related to this Light Fitting. Not required if the Light Fitting does not have any visual effect objects.
soundTag	String	The tag of all the sound objects related to this Light Fitting. Not required if the Light Fitting does not have any sound objects.
vfxBlueprint	String	The blueprint resref of the visual effect objects related to this Light Fitting. Not required if the Light Fitting does not have any visual effect objects.
onBlueprint	String	This is used together with “offBlueprint” when one is used the other is required. Adding these sets a Light Fitting to be a dual state light which spawns different placeables for on and off states. This should be the blueprint resref for the “on” placeable.
offBlueprint	String	This is used together with “onBlueprint” when one is used the other is required. Adding these sets a Light Fitting to be a dual state light which spawns different placeables for on and off states. This should be the blueprint resref for the “off” placeable.
startState	String	When you omit the variables “onHours” and “offHours” on a Light Fitting its default state becomes permanently on. You can override this to the opposite by setting the variable “startState” to the string “off”.
chainDelay	Float	How long (in seconds) a Light Fitting should turn on and wait before sending a chain event to the next light in the chain.
oneShotDelay	Float	How long (in seconds) a Light Fittings should turn on for when it receives a one shot event. Light Fittings which don’t have this variable will ignore one shot events.

Extras

There are a few miscellaneous extra included with SLS2, the Rod of Light item, a Sundial placeable and the NPC Time Girl.

Rod of Light

The Rod of Light is an item (resref: sls2_lightrod). This is designed for use by builders or by DMs. Just add it to your quickbar and then use it on yourself, you'll get a conversation with various options to change the lighting. This can be used to test out your lighting effects or to set certain light for roleplay purposes.

Sundial

The SLS2 sundial is a standard sundial with an "On Used" script added. Place one in your module and it will print out the current time when used.

Time Girl

Time Girl is a helper NPC you can place in your module for testing. She'll offer you many options to change the time when you talk to her.