==========
ShadowGear Development - Cybermancy Teleporter v1.1 - About
==========

Change Log

♦ November 29, 2013

- Initial Release

Key Features

♦ Highly configurable 'SkyNet'* teleporter engine.

- Ultra-simple rez-and-go setup.
- Destinations are pre-loaded so you never have to wait for a menu.
- Destination menu is bypassed if there is only one destination.
- Teleporters link together automatically and can be moved around with no need to reset them.
- Unlimited networks featuring collaboration support, sub-networking, and gateways.
- Supports up to 33 teleporters per sub-network.
- Intelligent, low-lag teleport algorithm using llSetRegionPos.
- Can teleport to and from anywhere on the same sim, up to 4096m in the sky.
- Menu-driven network configuration with text-input boxes via llTextBox.
- Drop-in sound effect and animation customization with changeable sound effect volume.
- Advanced, destination-aware access controls with four access modes.
- Notecard-based access/ban list supports up to 40 names.
- Link-Message based plugin API with sample script for custom special effects and features.
- Clear, detailed instruction manuals covering every feature.
- Configurable hovertext.
- Special unicode character support.
- ... and much more!

♦ Custom-made particles and sound effects on both departure and incoming teleport.
♦ Outer gear rotates and glows brightly when teleporter is in use.
♦ 11 color presets and 5 emblem presets via the addons menu.
♦ Fully modifiable with template for custom emblems included.
♦ Particle color follows the emblem color.

*Robot apocalypse not included.

Credits / Thanks

♦ Rayne Keynes for her help with the particle effects and for providing placeholder sound effects while this teleporter was still under development.

♦ My friends Aleyn, Roselee, and Zen for helping me beta test these by using them on their land and for putting up with almost daily updates in some cases. Extra thanks to Roselee and Mandy for their willingness to proofread and give feedback on the documentation for the SkyNet system. You guys rock!

♦ The people at www.obsidiandawn.com for providing the awesome brush set used to make the emblems.

♦ Prefabrica of SL, for making the free sculpted gear used in this design.

♦ Freesound.org, for providing royalty-free samples to work with when creating the sound effects.

♦ My friends and family at The Knot for providing me with a much needed place to relax after long hours spent writing code.

♦ And to my discerning customers, for choosing this product over the countless others in the vast metaverse that is SL. May it serve you well.

ShadowGear Development - SkyNet v2.4.1 - User Manual

1. The Basics

1.1. Basic Setup

Simply rez a teleporter and enter a short name for the location the teleporter will represent into the text input box that appears, then click 'Submit'. Repeat this step for each teleporter you need, up to 33. The teleporters will automatically locate each other and can be moved anywhere in the same region without further configuration.

This quick and simple setup process can be leveraged for more highly customized networks, as long as all settings but the teleporter name will be the same for a given group of teleporters. Simply rez and configure one teleporter, take it back into inventory, then rez as many copies of it as you need, following the quick setup instructions given above.

1.2. Basic Customization

The Animation and Sound Effect of the teleporter can be changed by dropping a new animation or sound into the teleporter's contents. To ensure the new animation or sound is used, remove any existing animations or sounds from the teleporter's contents first. To do this, right click the teleporter and select 'Edit', then navigate to the contents tab of the build/edit dialog.

2. Teleporter Configuration

2.1. Entering Configuration Mode

To access configuration mode, right click the teleporter and select 'Touch' from the menu. The teleporter will be placed into configuration mode and the configuration menu will be displayed. While in configuration mode, no-one can use the teleporter, and it will not respond to requests from other teleporters.

If you accidentally lose the configuration menu, or dismiss it with ignore, you can get it back by touching the teleporter. After 1 minute of inactivity, the configuration menu times out, and any settings you changed via the configuration menu will automatically be saved before the teleporter returns to normal operation.

2.2. The Configuration Menu

2.2.1. ✣ Locking

The teleporter has two independent locking options, which control whether it will allow incoming or outgoing teleports. Locks affect all users, including the owner, and can be easily toggled by addons. For more information about addons, see the Designer Manual.

A) Lock Incoming - If checked, all requests to teleport in from other locations will be denied.

B) Lock Outgoing - If checked, the teleporter is effectively in a 'disabled' state and will deny access to anyone trying to use it.

2.2.2. ♥ Access

The teleporter can be put into one of four different access modes, which restrict who can teleport to and from its location via the network. If an avatar is excluded from access by the mode specified, they will be shown a message indicating the nature of the access restriction.

A) Open Access - Anyone can teleport to and from this location via the network. (default)

B) Group-Only Access - Avatars teleporting to and from this location via the network must have the same active group as the group the teleporter is set to. To set the group for a teleporter, right click the teleporter and select 'Edit'. Then, on the 'General' tab of the build/edit dialog, select the small wrench icon to the right of 'Group', and select the group you want the teleporter to be set to. NEVER under any circumstances -deed- a teleporter to a group, as this will cause the configuration menu to stop functioning.

C) Access List - Avatars teleporting to and from this location via the network must have their names listed in the Access Notecard. See Section 3 for details on how to use the access notecard. If the access notecard is empty, this setting makes the teleporter effectively owner-only.

D) Ban List - Avatars teleporting to and from this location via the network must NOT have their names listed in the Access Notecard. As before, see Section 3 for details on how to use the access notecard.

2.2.3. 🏛 Boundaries

Network boundaries allow you to restrict which locations within the same network are added to a given teleporter's destination list. This allows for network overlap among different builders within the same region, and allows the creation of very complex networks. See Section 4 for more details on complex networks.

A) Bounding by Owner - If checked (default), the teleporter will not acknowledge any destinations that are not owned by you. If unchecked, enables collaboration between multiple builders. See Section 4.2 for details.

B) Bounding by Subnet - If checked, the teleporter will not acknowledge any destinations that have a different sub-network name. This primarily allows for complex networking. See Section 2.2.4. for details on how to set the sub-network name, and Section 4.2 for details on complex networking.

2.2.4. ✎ Rename

The rename menu allows you to set the three components of a teleporter's 'network address', which uniquely identify it to other teleporters in the same region. They are limited to 21 bytes each, and support unicode characters and symbols. If the text you enter into the text input box is longer than 21 bytes, it will automatically be truncated.

A) ✳ Teleporter - This specifies the unique, short name of the teleporter itself. Usually this is set to a short name for the location the teleporter represents, such as 'home', or 'main store'. If the name is not unique, it will be prefixed with a random number by any other teleporters on the network.

B) ☎ Network - This specifies the name of the network. The name is case sensitive, and a teleporter can only communicate with other teleporters that have the same network name.

C) ☂ Subnet - This specifies the name of the sub-network. This name, like the network name, is case sensitive. The sub-network name is only used if the network is bounded by subnet. See Section 2.2.3 for details on network boundaries.

2.2.5. ⓣ Hovertext

The hovertext option displays a text input box allowing you to set the floating text displayed when the teleporter is idle. By default, this is 'Teleporter Ready'. This field is limited to 21 bytes, and supports unicode characters and symbols. If the text you enter is longer than 21 bytes, it is automatically truncated.

2.2.6. ↕ Volume

The volume menu allows you to change the volume of any sound effects played by the teleporter, and to preview the sound effect at the volume you set. By default, the volume is 100%.

A) ▲ x% - These buttons shift the volume of the sound effect up by the percentage indicated.

B) ▼ x% - These buttons shift the volume of the sound effect down by the percentage indicated.

C) ♪ Preview - This button triggers the teleporter's 'incoming teleport' effect, which plays the sound effect at the volume indicated, effectively allowing you to preview it. If no sound effect is available to preview, it will display a message instead.

2.2.7. ✯ Addons

If the designer of your teleporter provided any menu-configurable addons, such as texture or color change, they will appear here as options. If not, clicking this option should display a message and return you to the main menu.

2.2.8. ☯ Memory

This option displays the amount of memory remaining, in bytes and as a percentage. Things which can affect free memory include adding more teleporters to the network, using longer names, and adding names to the access notecard.

2.2.9. ✔ Save

When clicked, any configuration changes are saved and the teleporter is returned to normal operation, where it immediately resynchronizes with any other teleporters on the network.

3. Restricting Access

3.1. The Access Notecard

The access notecard should be located in the contents of the teleporter, and is named '_access'. To use it, right click and select 'Edit' from the menu, then in the build/edit dialog, go to the 'Contents' tab, and double-click the notecard to open it. If it does not open right away, select some other object, then re-select the teleporter and try again.

Lines which begin with a hashtag (#) are comments, which are ignored when the notecard is read. Enter the legacy names of avatars you want to allow/deny access to, one name per line. For newer avatars, the legacy name will end with 'Resident' (ex. JaneDoe Resident). Do not enter display names. They are not unique and the access system cannot recognize an avatar by its display name.

You can add up to 40 names to the access notecard. If you add more than this, only the first 40 will be read. Once you save the notecard, the teleporter will automatically restart in order to load the changes.

3.2. How the Access Notecard is Used

The behavior of the access notecard is defined by the teleporter's access mode. For more information on setting the access mode, see Section 2.2.2. For Open and Group-Only access modes, the teleporter does not use the access notecard, though it will still load it. In Access List mode, the teleporter will allow access only to those avatars whose names appear in the access notecard. In Ban List mode, however, the teleporter will deny access to avatars whose names appear in the access notecard.

4. Advanced Networking Techniques

4.1. Cloning Teleporter Settings

Advanced users may back up and clone the settings of any teleporter. To clone the settings of a teleporter, copy its description. It looks like gibberish, but it contains all the settings of that teleporter, except for the contents of the access notecard.

Then, on the teleporter you want to clone the settings to, enter configuration mode and open any menu option that uses a text input box, such as 'hovertext', and paste the copied description into it. Click 'Submit', and the teleporter should read and apply the settings, then restart.

4.2. Collaboration / Multi-Owner Networks

There are several ways to create collaborative or multi-owner networks. The simplest method is to create the network with the 'owner' network boundary disabled. See Section 2.2.3 for more information on setting network boundaries. Be sure to change the network name from its default if your group wants to use this feature. A collaborative network in this configuration will support 33 teleporters, just like the basic network in Section 1.1.

If your group needs more than 33 teleporters, or if something more complex is desired, you can set up the collaborative network using sub-networks and gateways as outlined in Section 4.3, below. Both network boundaries from Section 2.2.3 work in the same general way, and can be used together or alone. Combined with the access control features, this gives SkyNet systems the potential for a great deal of complexity.

4.3. Sub-Networking / Gateways

4.3.1. Sub-Networks

A sub-network, as defined within the context of a SkyNet system, is a group of teleporters on the same network which communicate with each other, but not to other teleporters within the network. Sub-networking is controlled by the network boundaries feature outlined in Section 2.2.3. By default, a teleporter is bounded by owner, which automatically puts it into an owner-based sub-network. If it were not, anyone else on the region who was using the default, easy-setup configuration would have their teleporters show up on your network, and vice versa.

The other network boundary, 'subnet', uses the subnet name as the boundary, a case-sensitive, settable property located in the rename section of the configuration menu. This allows sub-networking within networks which have the same owner, rather than restricting the ability to collaborative networks.

Simply enabling a network boundary will create a sub-network. However, a sub-network will behave in almost exactly the same way as the network itself by default. That is, it will exclude destinations which fall outside of the network boundaries defined for the teleporter. To make sub-networks a bit more useful, one must define gateways. See Section 4.3.2 below.

4.3.2. Gateways

A gateway, as defined within the context of a SkyNet system, is a teleporter which acts as a bridge between two or more sub-networks. Each sub-network to be joined must have at least one gateway. To define a teleporter as a gateway, simply remove the network boundary restriction(s) being used to define the sub-networks on that teleporter, but leave the boundaries in place for all the other teleporters. Once you have done this, the gateway will still be able to communicate with all of the teleporters on its native sub-network, but it will also be able to communicate with the gateways of other sub-networks.

Once there is a gateway in each sub-network being bridged, avatars will be able to teleport to the other

sub-networks by first teleporting to the gateway. Sub-networks, with the addition of gateways, allows one to create linked networks with more than 33 teleporters, as well as create more complex network layouts for routing avatars where you want them to go. If you plan to create a complex network such as this, it is useful to sketch a rough network diagram to get an idea of how avatars will be routed within a given network.

As an example, suppose you had two sub-networks bounded by subnet. To bridge them you would create a gateway in each sub-network by removing the 'subnet' network boundary for the two gateway teleporters. This would define them as gateways between sub-networks that are bounded by subnet. This is independent of the owner-based network boundary, so you could have a completely different set of gateways along that boundary if desired.

Also note that when gateways are in use, a sub-network loses one teleporter capacity for each link created by the gateway. So, in the example above, the capacity for each of the two sub-networks would decrease by one, since the gateways only create a single link. The new capacity of each sub-network would then be 32, including the gateway itself, for a total of 64 reachable destinations.

5. Frequently Asked Questions

Q) Where is the 'Owner Only' access restriction? I can't find it!

A) As stated in Section 2.2.2, you can make a teleporter/destination owner-only by setting the access type to 'Access List' and not adding any names to the access notecard. An empty access notecard in access list mode is essentially saying 'nobody is allowed', except the owner, who can use the teleporter regardless of the access mode.

Q) I see random numbers in front of some of my destination names in the menus of some teleporters. What's going on?

A) Some of the destination names on your network are not unique. To ensure each destination can be reached, the teleporters prefix non-unique names with a random number. To stop this from happening, rename the affected teleporters to something more unique. See Section 2.3.4 for instructions.

Q) If I delete or change the network a teleporter is on, it doesn't immediately disappear from the destination menu. Why?

A) To ensure low lag, the teleporters don't continuously poll for potential destinations. Since a deleted or re-networked teleporter cannot tell its former comrades it's gone, its name will continue to appear in menus until the network is resynchronized, or until an avatar attempts to teleport to it, at which point the problem will self-correct. If this is bothersome, you can force the network to resynchronize by entering the configuration menu of any teleporter on the network and then clicking 'Save' without changing anything.

Q) Sometimes, if I try to use or configure the teleporter right after it finishes booting/syncing, the action times out before I can do anything. What's happening?

A) Because of the way the timer event works, sometimes a stray event can get stuck in the queue as the teleporter changes states. There seems to be no clear way to keep this from happening, so our best suggestion is to make sure you wait for the idle/ready hovertext to appear before trying to perform any actions on the teleporter.

Q) My teleporters are resynchronizing at unexpected times. What's happening?

A) There is probably another avatar in the region configuring a SkyNet based teleporter system on the same network. This should honestly not happen very often. If it continues to happen and it bothers you, you can try switching to a different network name. See Section 2.2.4 for details.

Q) I think I found a bug / I have a problem not addressed here. What can I do?

A) If you're having problems while using the SkyNet system, drop Alex Carpenter a notecard with a detailed description of what happens, what you were doing at the time, the conditions on the sim, and any details about your configuration that you can remember. He'll get back to you as soon as possible, usually within 24 hours.

Q) Why doesn't SkyNet have xxx feature? Can I request it?

A) Feature requests are welcome. In some cases it is possible the feature was considered but wasn't added in order to maintain the efficiency of the core script. Additionally, if you have scripting skills, you might want to take a look at the designer documentation, as you may be able to write an Addon to add the functionality that you want. If you'd still like to request a feature, you can do so by dropping Alex Carpenter a notecard. He'll get back to you as soon as possible, usually within 24 hours.

1. Basic Design Principles

1.1. Setting Up Your Build

SkyNet can be used to convert almost any prim or linkset into a fully functioning teleporter. All of the functionality described in the User Manual is included as part of the base system. If you have a particular prim you want the system's floating text to appear over, edit linked parts and select that prim, then name it 'CHASSIS', in all caps, without the single quotes. The base system will scan for this prim on startup and set the floating text on that prim. Make sure only ONE prim is using this name.

In the event floating text is displaying on an undesirable prim, change the name of any prim that you named 'CHASSIS' back to 'Object', or any other name, really. Then create a new script inside the object and insert the following line into the 'state_entry' event:

```
llSetLinkPrimitiveParams(LINK_SET, [PRIM_TEXT, "", ZERO_VECTOR, 0.0]);
```

Save the script. Once it has run, the floating text should disappear. Delete the script you just created, then follow the instructions in the first paragraph of this section. After you have done this, reset the Base System script.

1.2. Installing SkyNet

Once your teleporter design is set up the way you want, edit the object and navigate to the Contents tab of the root prim (this is selected by default if Edit Linked Parts is not checked). Then, drag a copy of the '_access' notecard into the contents tab first. Then, drag in a copy of the Base System script. Your new teleporter should immediately start up.

The script will probably whisper a warning about missing configuration. This is normal. The script will automatically initialize and save some default settings to the description of the root prim. If for some reason you receive a warning about a missing access notecard, simply drag the notecard named '_access' into the contents of the same prim the script is in, and the script should recognize it.

Optionally, if you have an animation you want to use, drag it into the contents tab with the script as well. SkyNet will work perfectly fine without one, but it can be useful if you're going for a particular effect. In the same way, you can also add a sound effect, with no scripting required. However, to get the most out of these effects, you will probably want to create a basic addon, which will allow you to set the sit target, set the sound by UUID, define special prim effects, and more. See Section 2, below.

2. Addon Architecture

2.1. Introduction

SkyNet supports a very basic Addon architecture, which allows other scripts to respond to and control some of the system's behavior. Messages are passed asynchronously between the Base System and any addon scripts in the same linkset, via the following command syntax:

```
llMessageLinked(LINK_SET, 9000, <COMMAND>, <ARGUMENTS>);
```

Note all messages use a 'number' of 9000 to differentiate them with any link messages you may desire between your own scripts inside the object. There are two types of messages within the SkyNet Addon API. Commands prefixed with 'FX_' are outgoing messages intended for use by addons. Commands prefixed with 'TP_' are requests the Base System is listening for from addons.

2.2. Assumptions

There are two commands which MUST be handled when you create an addon script. For more information on these commands, see Section 2.3. Additionally, a skeleton addon script should have been included with this package.

FX_INIT - If you set a custom sit target, or send the 'SETSOUND' or 'SETDELAY' commands, you should either reset your addon script in response to this command, or at the very least reset the sit target, and resend any of the above two commands your script uses.

FX_MENU - The Base System assumes at least one addon script will handle this command if another script is present in the same prim. Even if you don't plan to have menu options, you should handle this event in some way, such as issuing the 'TP_MENU' command. This ensures the user cannot 'fall out' of the menu by clicking the Addons option.

2.3. Teleporter Events

Teleporter Events are commands sent by the Base System when it changes states, in order to allow addon scripts the ability to respond to these state changes as well. This is useful for keeping addons in sync with the base system, as well as for triggering special effects. They are not available when the teleporter is in configuration mode. The events available to addons are as follows:

FX_ENABLE - Sent when the system is switched to an 'enabled' state, either by the user or by an addon.

FX_DISABLE - Sent when the system is switched to a 'disabled' state, either by the user of by an addon.

FX_INIT - Sent whenever the system is fully loaded and beginning to search for other devices on its network. This happens right after the device boots, when the user exits configuration mode, or when it is resynchronizing with other devices.

FX_STOP - Sent to indicate any active tasks are now complete and any effects related to those tasks should stop. Signifies the transition back to an idle state.

FX_CANCEL - Same as FX_STOP, except it is sent only when a teleport requested by a user has failed, such as when the access controls of the destination have denied the user permission to teleport in.

FX_START - Sent when a user sits on the teleport pad, signifying the beginning of a new teleport task.

FX_DEPART - Sent when a request to teleport succeeds, just before the device moves the user to the destination.

FX_RETURN - Sent immediately after the device drops the user off at their destination, signaling it is about to return to its home location.

FX_INBOUND - Sent when another device has requested to teleport a user to the teleporter's home location, and the user has passed all access checks.

2.4. Teleporter Commands

Teleporter Commands are commands which are sent by addon scripts to control the behavior of the Base System. These include basic control functions and provisions to set certain internal properties, mainly for the purpose of effects customization. They are not available while the teleporter is in configuration mode. The commands available to addons are as follows:

TP_ENABLE - Signals the system to switch to the 'enabled' state. Takes no arguments.

TP_DISABLE - Signals the system to switch to the 'disabled' state. Takes no arguments.

TP_UNLOCK - Signals the system to allow teleports from other destinations. Takes no arguments.

TP_LOCK - Signals the system to refuse teleports from other destinations. Takes no arguments.

TP_REBOOT - Signals the system to perform a complete reboot. Takes no arguments. This command resets the script, forcing the system to reload all settings. Also triggers the FX_INIT event.

TP_SETSOUND - Given the UUID of a sound as an argument, signals the system to set the given sound as the default teleport sound effect. This sound can still be overridden by the user via dropping a sound into inventory. Must be resent if the base system script is reset.

TP_SETDELAY - Given a floating point/decimal number as an argument, representing a time, in seconds, sets the departure delay to the number of seconds indicated. The departure delay is used during departure and incoming teleport events, and is meant to gives any special effects the designer creates time to fire before the teleport sequence continues. The default delay is 0.1 seconds. Must be resent if the base system script is reset.

2.5. Configuration Mode Commands

These are commands which are used only when the teleporter is in configuration mode. These commands control how and when menus are displayed to the user, and as such there is a tighter interplay between the commands sent and received by the base system. The commands available during configuration mode are as follows:

FX_MENU - Sent by the system when the owner clicks the 'Addons' button in the configuration menu. If an addon script is present, it must respond to this command, otherwise the user can 'fall out' of the configuration menu.

FX_MENUFWD - If the menu handler doesn't recognize a button that was clicked by the user, it will send this command, with the full name of the button the user clicked as its argument. This allows the addon script to handle its own menu buttons without requiring a separate listen handle.

TP_MENU - Requests the main configuration menu. If the system is not in configuration mode when it receives this command, it will enter configuration mode. Take no arguments.

TP_SHOWMENU - Given a special string as an argument, displays the menu described by the string to the owner. All of the usual caveats associated with llDialog still apply, such as the 12-button limit. The syntax for the command string is as follows:

<prompt>~|~<Button1>|<Button2|...|<Button12>

<Prompt> is the message you want to display above the buttons. <Button1> through <Button12> are of course the text to display on each button. Note the separator between the prompt and the button list is a pipe character with a tilde character on each side (~|~), and the button list is separated by pipe characters (|).

As a final note on the menu system in place during configuration mode, button names for addon functions must be globally unique. They will not be triggered properly if a button name matches one found anywhere in the main configuration menu tree. This also means you may leverage button names found elsewhere in the global tree if it is useful to you. For instance, using the '↑' symbol to allow the user to jump back to the main menu without the need to handle it in your addon.

3. Anatomy of a Settings String

3.1. Reading the Settings String

The settings string is an encoded list of settings stored in a teleporter's description. It consists of four strings and four integers, separated by newline characters (\n) and encoded in base64. To read it, you will

need to decode it back into a list, which can be done with the following LSL statement:

```
llParseStringKeepNulls(llBase64ToString(llGetObjectDesc()),["\n"],[]);
```

The statement above, assuming your script is in the root prim of the teleporter, will fetch the settings string, decode it, and convert it into LSL's list datatype. The contents of the list are in a specific order, and that order must be maintained for the settings to function properly. Their order and a brief description of each setting can be seen below:

0) Network Name - (string) The name of the network the teleporter is in. Limited to 21 bytes.

1) Subnet Name - (string) The name of the subnet the teleporter is in. Not used if Bound By Subnet is false. Limited to 21 bytes.

2) Teleporter Name - (string) The name the teleporter uses to identify itself to the network. Limited to 21 bytes.

3) Hovertext - (string) The floating text shown when the teleporter is idle. Limited to 21 bytes.

4) Sound Volume - (integer) Number ranged 0-100 indicating the sound volume, where 100 is equivalent to 100 percent (%).

5) Flags - (integer) A bitfield containing 5 settings, stored in binary form. Below is a list of the settings, from highest order bits to lowest. The two numbers before each setting are the bitmasks for AND and OR, respectively. The final setting takes two bits to represent, and therefore must be treated differently, hence the odd numbers.

[AND / OR]	[Description]
[31 / 32]	True = Lock incoming teleports.
[47 / 16]	True = Lock outgoing teleports.
[55 / 8]	True = Bound network by subnet name.
[59 / 4]	True = Bound network by owner.
[60 / 3]	0 = Open Access. 1 = Group-Only Access. 2 = Access List. 3 = Ban List.

Use the following guide for setting, clearing, toggling, and testing individual bits.

[Action]	[Operator]	[Mask]	[Description]
Test	AND (&)	OR	Test if setting is true.
Clear	AND (&)	AND	Set setting to false.
Set	OR (|)	OR	Set setting to true.
Toggle	XOR (^)	OR	Toggle setting.

For the access type setting, use the following syntax examples for testing and setting. Note that the AND mask, in this case, is used to clear the setting every time it is to be set to a different value. This ensures the correct type is set without affecting the other bits in the field. Additionally, the OR mask results in a number between 0 and 3, instead of true or false.

```
Test:  (flags & 3) == <setting (0-3)>
Set:   flags = (flags & 60) | <setting (0-3)>
```

3.2. Altering the Settings String

It is generally not recommended to change the settings of the device inside an addon script unless it is absolutely necessary. If this is the case, one can fairly easily make changes to the settings string once it

has been read using the method above. The quickest way to change a setting is to use the following LSL statement, which will find and replace a setting by its index in the ordered list 'mySettings':

```
mySettings = llListReplaceList(mySettings, [<new value>], <index>, <index>);
```

Note that <index> represents the index of the setting being changed in the ordered list. This can be seen in the numbers to the left of each setting in Section 3.1. <New value>, of course, is the new value for the setting indicated by the index.

Once you're done making changes to the settings string, use the following LSL statement to save them back into the teleporter's description. Note that it is extremely important that the settings remain in the same order they were originally in, which it should if you're using the recommended method. Also note that the string length limitations for the first four settings must be followed, or the settings may not save correctly.

```
llSetObjectDesc(llStringToBase64(llDumpList2String(mySettings, "\n")));
```

Once saved, you will need to trigger a reboot by using the 'TP_REBOOT' command from Section 2.4 for the settings to take effect. Ideally, this should be done immediately after the settings are saved, to avoid any potential for them to be overwritten by the base system itself.

4. Questions and Contact

If you have any questions that aren't addressed in the manual(s), please send a notecard to Alex Carpenter. He will get back to you as soon as possible, usually within 24 hours.