```
//////////////////////////////////////////////////////////////////////
// Simple owner-only RLV attachment locker with show and hide.
//
// Simple script to lock an attachment on an avatar, using RLV. This is owner-only, and uses voice commands
// It's useful to stop something being knocked off during outfit changes.
//
// This version does not use the State Machine so that the listener does not get reset on state change, it also reduces
// duplicate code. I have found it to be more flexible and reliable when dropping into HUDs and the like.
//
/// @author Rebecca Ashbourne
/// @version 2.11
/// @copyright 2017 Rebecca Ashbourne
//
// This script is free software: you can redistribute it and/or
// modify it under the terms of the GNU Lesser General Public License
// as published by the Free Software Foundation, either version 2.1
// of the License, or (at your option) any later version.
//
// This script is distributed in the hope that it will be useful, but
// WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
// See the GNU Lesser General Public License
// at http://www.gnu.org/licenses/ for more details.

// Settings
integer LISTENER_CHANNEL = 44;  // change this value to listen on a different channel
integer ALLOW_HIDE = TRUE;      // change this to FALSE to disable the show/hide functionality
integer isLocked = TRUE;        // change this to change the initial state
integer isVisible = TRUE;       // change this to change the initial state. If you set ALLOW_HIDE to FALSE then ensure this is TRUE.

// Global variables
integer listenerHandle;
integer beQuiet = FALSE;       // used to suppress output on initialisation
integer allowDebug = FALSE;    // set this to TRUE to allow debug display

//////////////////////////////////////////////////////////////////////
// Used for debug
debug(string msg)
{
    if (allowDebug)
        llOwnerSay("DEBUG: " + msg);
}

//////////////////////////////////////////////////////////////////////
// simple conversion of boolean integer to a string
string bool2string(integer val)
{
    if (val == TRUE)
        return "TRUE";
    else if (val == FALSE)
        return "FALSE";
    else
        return "(NaB)";
}

//////////////////////////////////////////////////////////////////////
// Sets the locked / unlocked state
setLocked(integer val)
{
    if (val)
    {
        llOwnerSay("@detach=n");
        isLocked = TRUE;
    }
    else
    {
        llOwnerSay("@detach=y");
        isLocked = FALSE;
    }
    if (!beQuiet)
        sayLockState();
}

//////////////////////////////////////////////////////////////////////
// Reports the locked / unlocked state
sayLockState()
{
    if (isLocked)
        llOwnerSay("Locked");
    else
        llOwnerSay("Unlocked");
}

//////////////////////////////////////////////////////////////////////
// Sets the visibility
setVisible(integer val)
{
    if (val)
    {
        llSetLinkAlpha(LINK_SET,1,ALL_SIDES);
        isVisible = TRUE;
    }
    else if (ALLOW_HIDE)
    {
        llSetLinkAlpha(LINK_SET,0,ALL_SIDES);
        isVisible = FALSE;
    }

    if (!beQuiet)
        sayVisibleState();
}

//////////////////////////////////////////////////////////////////////
```

```
// Reports the visibility state
sayVisibleState()
{
    if (isVisible)
        llOwnerSay("Show");
    else
        llOwnerSay("Hide");
}

////////////////////////////////////////////////////////////////////
// Ensure we are listening
doListen()
{
    listenerHandle = llListen(LISTENER_CHANNEL, "", llGetOwner(), "");
    debug("doListen(): Listening on channel " + (string)LISTENER_CHANNEL + " with listener id " + (string)listenerHandle);
}

////////////////////////////////////////////////////////////////////
// Say the status
showStatus(integer verbose)
{

    llOwnerSay("Listening on channel " + (string)LISTENER_CHANNEL + " with listener id " + (string)listenerHandle);

    if (verbose)
        llOwnerSay("Locked = " + bool2string(isLocked) + ", Visible = " + bool2string(isVisible));
}

////////////////////////////////////////////////////////////////////
// Initialise the object state, and reaffirm any lock
initialise()
{
    beQuiet = TRUE;
    debug("Initialising...");
    doListen();
    setLocked(isLocked);
    setVisible(isVisible);
    showStatus(FALSE);
    beQuiet = FALSE;
}

////////////////////////////////////////////////////////////////////
default
{
    ////////////////////////////////////////////////////////////////
    // State entry & exit
    state_entry()
    {
        debug("state_entry()");
        doListen();
        //initialise();
    }

    ////////////////////////////////////////////////////////////////
    // Triggered in an object when the object attaches or detaches from agent.
    attach(key id)
    {
        if (id)
        {
            debug("Attaching...");
            doListen();
            //initialise();
        }
        else
        {
            llOwnerSay("Detached");
            llListenRemove(listenerHandle);
        }
    }

    ////////////////////////////////////////////////////////////////
    // Respond to a voice command
    listen(integer LISTENER_CHANNEL, string name, key id, string message)
    {
        if (id == llGetOwner())
        {
            if (message == "lock")
            {
                setLocked(TRUE);
            }
            else if (message == "unlock")
            {
                setLocked(FALSE);
            }
            else if (message == "show")
            {
                setVisible(TRUE);
            }
            else if (message == "hide")
            {
                setVisible(FALSE);
            }
            else if (message == "status")
            {
                showStatus(TRUE);
            }
        }
        else
        {
            llOwnerSay("Not the owner");
        }
    }

    ////////////////////////////////////////////////////////////////
    // Triggered when an object is rezzed (by script or by user). Also triggered in attachments when a user logs in, or when the
```

```
    object is attached from inventory.
    // on_rez will be triggered prior to attach when attaching from inventory or during login.
    on_rez( integer _ )
    {
        debug("on_rez()");
        initialise();
    }
}
```