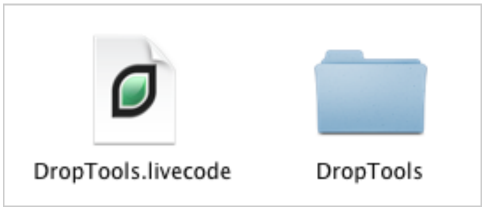
DropTools User Guide

Installation

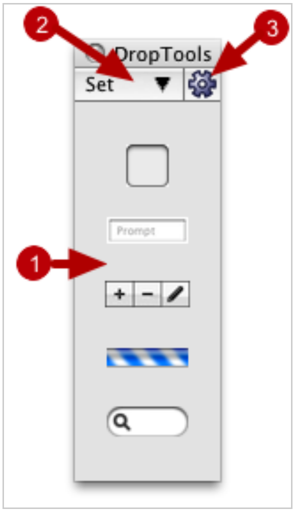


To install, place the **DropTools.livecode** file and the **DropTools** folder inside your LiveCode Plugins folder (wherever you have it). If you have not created a plugins folder for LiveCode, you will need to create one:

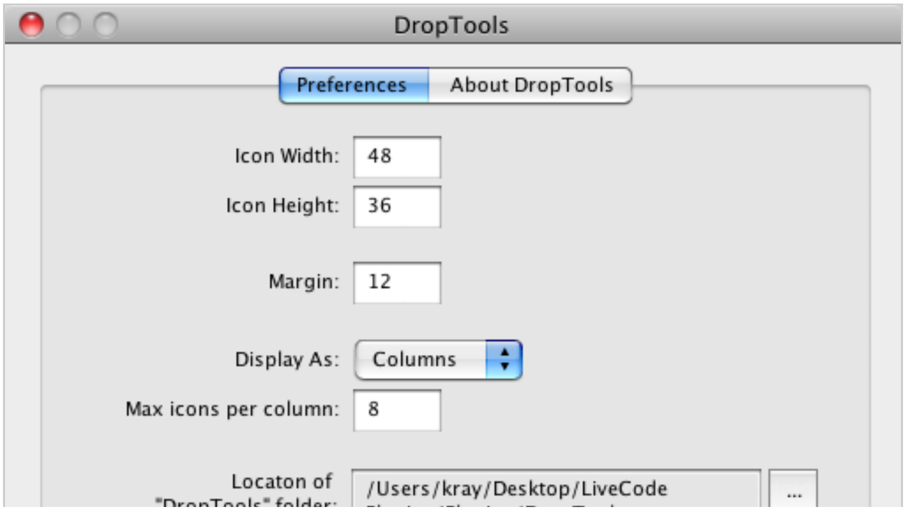
1. Create a folder for holding your LiveCode plugins (I named mine "LiveCode Plugins").
2. Inside the folder you just created, create a folder just called "Plugins".
3. Launch LiveCode, and select "Preferences" from the **Edit** menu (Windows/Linux) or **LiveCode** (application) menu (Mac OS X).
4. When the LiveCode Preferences window appears, click on "Files & Memory" from the list on the left.
5. At the bottom of the right-hand pane, there is a label called "User Extensions:" with an ellipsis ("...") button to the right of the field.
6. Click the ellipsis button and select the folder you created in Step #1 (<u>**not**</u> the "Plugins" folder you created in Step #2).
7. You should get a dialog box telling you the LiveCode must be restarted; so click "Quit" and relaunch LiveCode.

Using DropTools

Choose "stsDropTools" from the **Plugins** submenu of the **Development** menu in the LiveCode IDE (if you are using the MetaCard IDE, choose "DropTools.livecode" from the **Plugins** menu). This will display the **DropTools Palette**, displaying all of the custom controls ("DropTools") that are inside the **DropTools** folder.



The controls are loaded into the main area of the palette (1). If you have added subfolders into the **DropTools** folder (see "Organizing Your DropTools", below) the Sets menu (2) will let you navigate between the sets. Click the Options button (3) to set options for the **DropTools Palette**:



Icon Width: 48

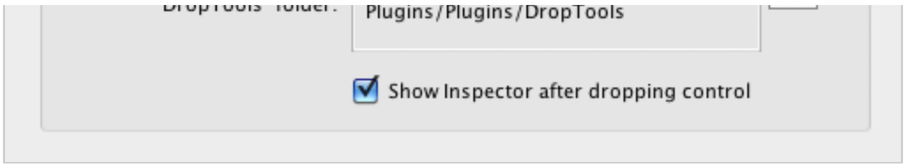
Icon Height: 36

Margin: 12

Display As: Columns

Max icons per column: 8

Locaton of "DropTools" folder: /Users/kray/Desktop/LiveCode

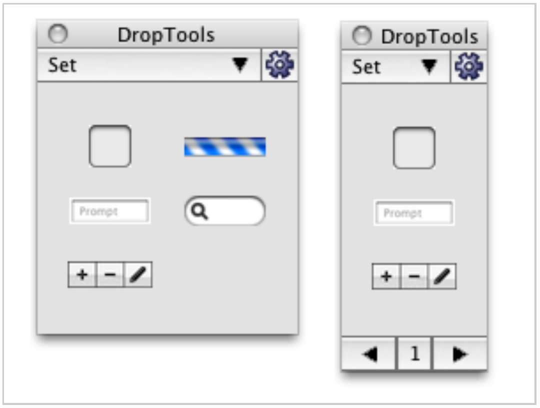

DropTools folder: Plugins/Plugins/DropTools

☑ Show Inspector after dropping control

Setting Options

You can set the **Icon Width** and **Icon Height**, which sets the width and height of the "slot" that displays the custom control icons. (The default width is larger than the default height because most custom controls seem to be horizontal in nature.) You can also set the all-around **Margin** distance (in pixels) around the icons.

The **Display As** popup menu, along with the **Max icons** setting lets you pick how you want the icons displayed. You can display the icons in snaking columns, or you can display the icons in "pages", which will display navigation buttons below the icons. Here's an example showing the difference, with five controls and a **Max icons** setting of 3:



You can set the location of where the **DropTools Palette** goes to look for DropTool stacks to load. It defaults to pointing to the "DropTools" folder in the same directory as the **Droptools.livecode** stack file, but you can move it wherever you like and then set the location in the Preferences window.

(NOTE: The path that is displayed is where the DropTools folder currently is; if you go to change it, you are picking the folder that will *contain* the DropTools folder. The DropTools Palette is smart about this, though, and will take the path that comes back from the dialog box, and see if it ends in "/DropTools"; if so, it will use that folder. However if you pick a folder that does not end with "/DropTools", it will add "/DropTools" to the end of the folder path.)

Finally, you can set the **Show Inspector after dropping control** checkbox. If this is checked, and the DropTool control that you drop onto your card contains its own Inspector (see Using an Inspector below), the Inspector will automatically be opened so you can set options for the control you just dropped.

Using Dropbox?

If you have multiple computers and a Dropbox account, you may want to consider putting the DropTools folder into your Dropbox folder and point all of your DropTools Palettes to that folder - this way, when you get a new DropTool, you can just add it in one place and all of your computers will get to use it immediately!

Organizing Your DropTools

By default, any DropTool stacks that are in the **DropTools** folder are displayed in the **DropTools Palette** whenever it is opened. As you add more and more DropTool stacks, you may find you want to be able to organize your DropTools into groups.

You can group your DropTool stacks by adding subfolders to the **DropTools** folder; the name of the subfolder(s) will show up under the "Sets" popup menu in the **DropTools Palette**. (Only one level of subfolders is supported.)

If you want the controls to show up in a different order, you can rename the stack files in the **DropTools** folder to get the order you want (controls are loaded alphabetically by file name). For example, in the screenshots above, the first three controls are "stsImageWell", "PromptField", and "stsGradientButton". Normally, this would mean the "stsImageWell" would be listed *last*. However in this case, the file name was "1-stsImageWell.livecode", so it loaded and displayed first.

Note: Only stack files with a .livecode, .rev, or .mc are recognized as legitimate DropTool stacks. Any other file extensions are ignored. Additionally, any folders that start with a "(" will be ignored when loading DropTools (folders that start with "(" are designed to be used as support folders for any DropTool that needs to maintain files on disk for its own purpose).

Using the Palette

To add a DropTool to your stack, click an drag from the **DropTools Palette** to your stack (the "drop stack"). When you do this, the DropTool will be added to the current card of the drop stack, and all the necessary supporting behaviors, resources, images, or other objects ("support resources"). will be installed for you.

By default, a stack named "DropTools_<milliseconds>" is created as a substack of the drop stack, a card is added to that substack for the kind of DropTool you dropped, and all the support resources are copied to that card. (You can change where the support resources are installed if you have a special place you want to keep them; see "Advanced Options", below.)

The substack and card will only be created if they don't already exist; this way, you can drop multiple instances of the same kind of DropTool on different cards of the drop stack, and there will still only be one card in the substack that controls them all.

Different Kinds of DropTools / How Each is Supported

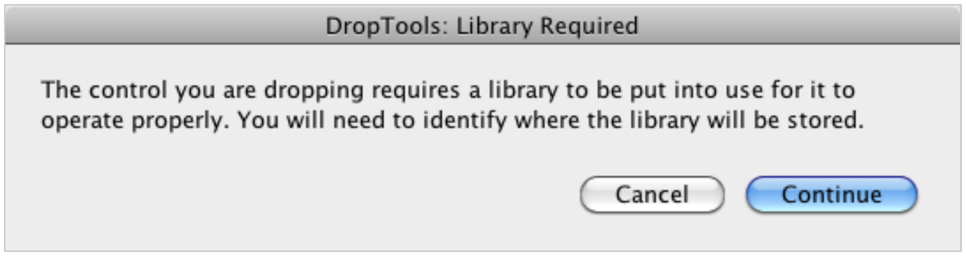
As of the date of this documentation, the DropTools Palette is designed to support three different kinds of DropTool controls:

1. **Prescripted:** The control has all the code built into it and is self-contained.
2. **Behavior-Based:** The control has little or no code of its own, but the scripts that drive it are located in a behavior button.
3. **Library-Based:** The control has little or no code of its own; the scripts that drives it is in a library stack.

In general, the sequence of events described above in "Using the Palette" are followed for **Prescripted** and **Behavior-Based** DropTools; **Library-Based** DropTools are handled a bit differently, since they require a library stack to be added and brought "into use".

Using Library-Based DropTools

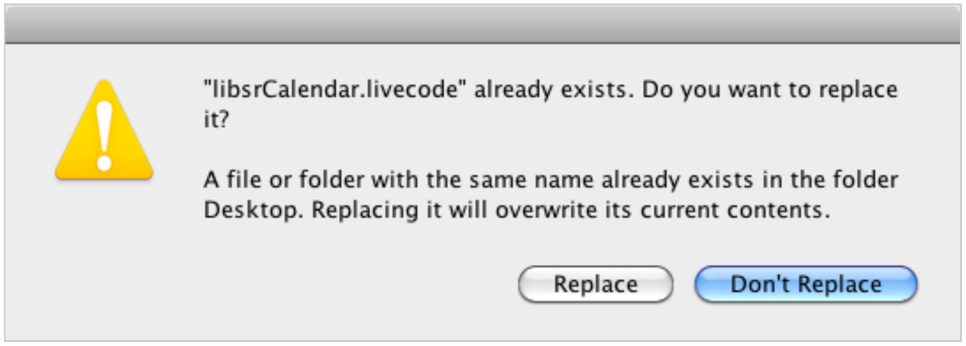
When you drag and drop a Library-Based DropTool to your stack, a test is made to determine if the library that it needs is already in use (which would be the case if you were to drop a second control onto your stack). If it's already in use, you won't see any prompts or dialogs. But if it is *not* in use, you will see this dialog box:



DropTools: Library Required

The control you are dropping requires a library to be put into use for it to operate properly. You will need to identify where the library will be stored.

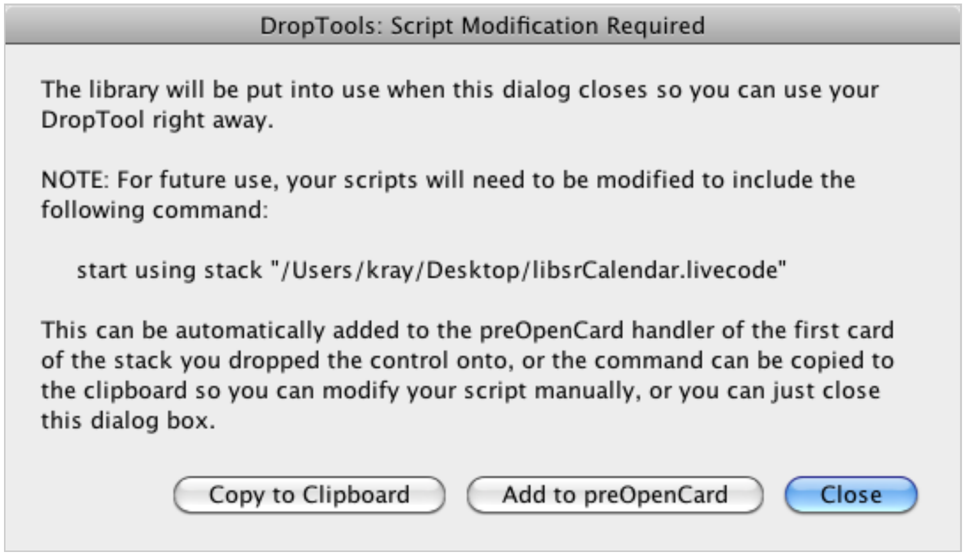
If you click "Continue", you will be given a standard dialog box to identify the folder where the library stack will be installed. If you select a folder that already has that same-named library file you will get this dialog:



"libsrCalendar.livecode" already exists. Do you want to replace it?

A file or folder with the same name already exists in the folder Desktop. Replacing it will overwrite its current contents.

The DropTools Palette will then install the library in the selected folder and deal with installing the rest of the support resources. Finally, it will bring the library stack "into use" and present this dialog box:



DropTools: Script Modification Required

The library will be put into use when this dialog closes so you can use your DropTool right away.

NOTE: For future use, your scripts will need to be modified to include the following command:

```
start using stack "/Users/kray/Desktop/libsrCalendar.livecode"
```

This can be automatically added to the preOpenCard handler of the first card of the stack you dropped the control onto, or the command can be copied to the clipboard so you can modify your script manually, or you can just close this dialog box.

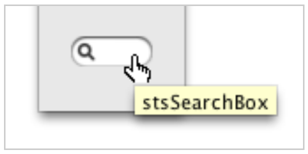
If you pick "Copy to Clipboard", you will be responsible for pasting the "start using" command into your stack in the appropriate place. If you want the command added automatically, click "Add to preOpenCard", which will either create a preOpenCard handler with the command in it, or it will insert the command as the first statement in a Preexisting preOpenCard handler. If you don't want anything done (you'll type it in yourself), click "Close".

Some Other Things You Might See

In order to try and accommodate a wide variety of needs by the DropTool developer to make sure their control gets used "correctly", you may see other dialog boxes or have additional actions that can occur after a control is dropped onto the drop stack. This is because the DropTools Palette sends certain messages during the loading and use of a DropTool that a DropTool developer can trap and take action on. For example, a DropTool may need you to select a file or copy in additional things manually, etc., which may cause you to see more than what is described in this manual.

Optional Features Provided by the DropTool Developer

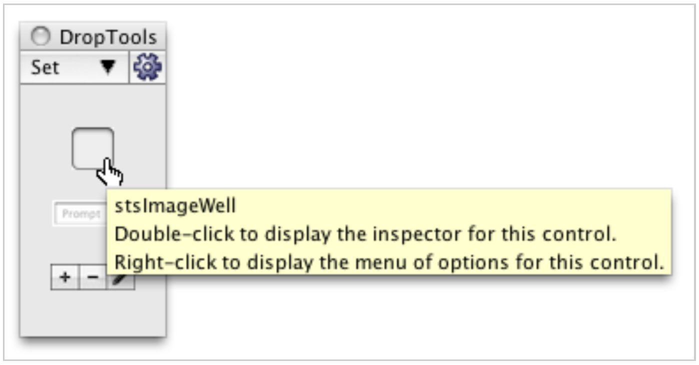
If you move the mouse over the controls in the DropTools Palette, you will see a tooltip that tells you what the DropTool is that you're pointing at:



There are some optional things that the DropTool developer can add to a "simple" DropTool to enhance the user experience and/or provide additional information. Currently those are:

1. An **Inspector** palette stack for setting properties or working with a selected instance of their DropTool;
2. An **About Box** stack which can include not only any "about" information the developer desires, but could also contain built-in help, links to examples, and so on.
3. Simple **about** information stored in certain custom properties that can provide some basic information about version, licensing, etc.

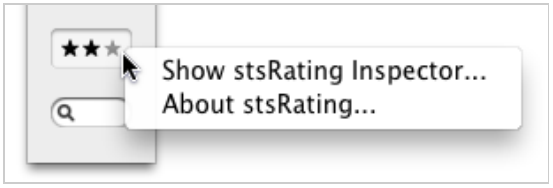
If the DropTool developer included any of the above with their DropTool stack, you will see "Right-click to display the menu of options for this control." displayed in the tooltip. If they included an Inspector palette, another line of text is added to the tooltip that reads: "Double-click to display the inspector for this control."


DropTools
Set
Prompt
stsImageWell
Double-click to display the inspector for this control.
Right-click to display the menu of options for this control.

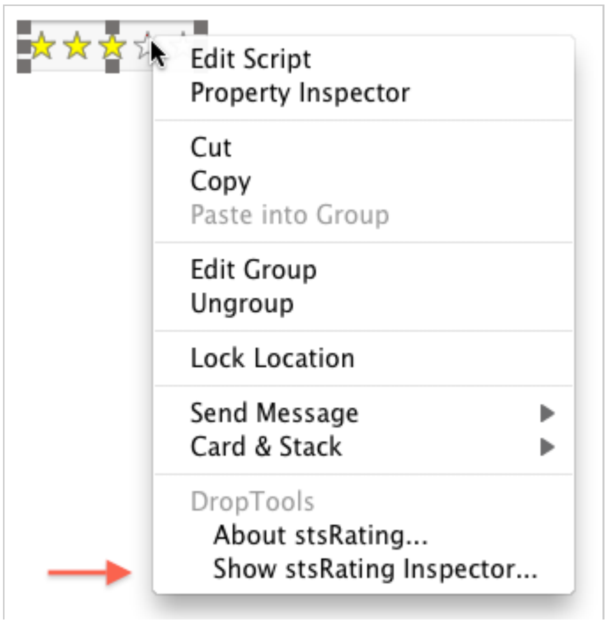
Using an Inspector

If the DropTool developer chose to include an Inspector, you can display it using any of the following methods:

- Double-click the corresponding DropTool icon in the DropTools palette.
- Right-clicking (or Control-clicking) the corresponding DropTool icon in the DropTools palette, and choosing "Show (DropToolName) Inspector..." from the dropdown menu.

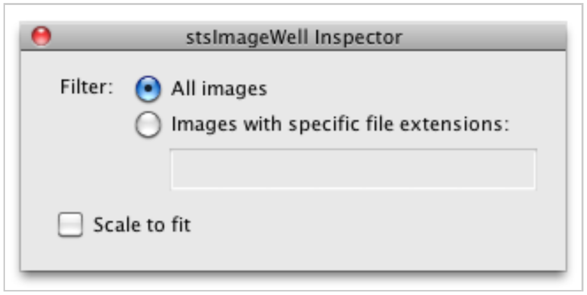


- Holding down the Option/Alt key and double-clicking the DropTool control on your card with the pointer tool.
- Right-click (or Control-click) the DropTool control on your card with the pointer tool, and choose "Show (DropToolName) Inspector..." from the bottom of the contextual menu that displays. This is an example of what you get with the stsRating DropTool:



Additionally, if the **Show Inspector after dropping control** checkbox is checked in Preferences, the Inspector should show up automatically after you have dragged and dropped the corresponding DropTool control onto your card.
Regardless of how you open the Inspector, it will be displayed as a floating palette, and it will let you adjust settings for the currently selected DropTool.

For example, take a look at the **stsImageWell**. It allows for images to be dragged and dropped onto it, and by default it accepts all image types (all those that LiveCode supports, that is). However you can set a custom property called "uSTSImageWell_Filter" that is a comma-delimited list of file name extensions to allow. You can set this manually if you like through the Custom Properties area of the LiveCode Inspector, but an easier way is to show the **stsImageWell Inspector**. Double-click on the **stsImageWell** control in the DropTools Palette, and you see a palette appear:



stsImageWell Inspector

Filter: [✓ All images] Images with specific file extensions:

☐ Scale to fit

Suppose you only wanted to accept PNG files; all you would need to do is make sure the stsImageWell control was selected, and add "png" into the **stsImageWell Inspector**. It automatically sets the property for you. The Inspector is also designed to disable if you have no control selected or if you don't have the correct type of control selected. Easy!

Advanced Options

Custom Install Location: If you don't want to accept the default installation location for support resources (a specific card of a substack of the drop stack), you can specify where you want these support resources to go. To do this you will need to set two things: a custom property (**uRIP["resourceCard"]**) of a stack, and a global variable (**gRIPAppStack**) specifying what stack has that custom property. Here's the technical details:

- The **gRIPAppStack** global property is the short name of a stack that is the mainstack of the application you're working on.
- The **uRIP["resourceCard"]** property of a stack can be either a card descriptor or a stack descriptor. If it is a card descriptor, then all support resource of *all* DropTools will go onto that one card. If it is a stack descriptor, then it will be treated the same way as the substack of the drop stack is (i.e. look for a card with the same type as the control being dropped, and if you don't find one then create a card for it), with the exception of course is that it's a different stack. Note that this stack descriptor can be either a mainstack or a substack (since it is only creating cards and copying objects).

Final Thoughts

If you run into any problems or have any suggestions, send an email to kray@sonsothunder.com.