

Windows > Actions

**Toolbox List Window:** Contains hierarchical list of ActionScript actions, operators, functions, properties, and objects.

**Actions List Window:** Contains the code of the ActionScript attached to the currently selected button, movie clip or frame. Each line represents a statement/action in the overall ActionScript. Selecting a statement by clicking it causes the Parameter pane to change, so that relevant parameters can be entered or edited.

**Parameter pane:** This is where you can enter or edit information pertaining to the currently selected statement. The pane is context sensitive and changes according to the type of statement selected in the Actions List window.

**Target Path Button:** Pressing this button opens the Insert Target Path dialog box, which lets you quickly choose a target for an action. This button is only activated if the parameter you are setting requires a target path.

Add/Delete buttons

Up/Down buttons

Options button: normal versus expert.

Expert Mode

The Actions list acts like a text editor. You can still add actions using the Toolbox list or the Add button, but you can't delete, change their sequence or configure their parameters as you would in Normal mode.

Choose your mode in Edit > Preferences. However Flash will remember in which mode you created a script.

You may add multiple actions to a single event, such as going to a specific frame and setting the transparency of a movie clip, or you could set it up so that different mouse events cause these same actions.

Nested Actions

Combining actions to establish a series of changes. For example, a Tell Target action allows you to set a target to perform an action; a secondary action must tell the target what to do. Using one action within another in this way is known as nesting. Nested actions are indented in the Actions List window.

Deleting Actions: select and click minus or delete

Pay attention to sequence of actions, for example tell target and then goto...

To reorder actions: in actions list window click once to select the action, and click the arrow buttons at the top of the actions panel to move the action up or down.

You may cut, copy and paste actions just as you would in an editor.

Basic Actions in Depth.

Syntax

If using an object action: mouse event e.g. on (release)

Curly Braces in ActionScript, JavaScript and many other languages enclose blocks of statements that belong together. Curly braces must always be used in matched pairs. You use curly braces most commonly in function definitions and control structures. The closing brace lets flash know that whatever statement may come next is a statement outside of the function definition.

Semicolon closes an action.

Expressions are based in numerical and mathematical values versus String Literals which must exist within quotes, because they are names that you may use within flash without a given value.

Basic actions go a long way, allowing you to create powerful interactivity with little effort. In most actions you need to evaluate expressions > You must set parameters for most of the actions in Flash to make them function the way you want. For example, the goto action requires you to

identify a frame number or label to go to. You may enter an exact value, such as 25, or you can set the value of the parameter based on the value an expression evaluates. In other words, say instead of entering a value of 25, you entered  $18 + 7$ , this is an expression, an equation or phrase that evaluates or equals a specific value, 25. Using this expression for the parameter's value would cause the goto action to go to frame 25 just as if you entered 25. Expressions can take many forms and can evaluate to many types of values, including numbers and text...the password is one example of using an expression or you may even use an expression with the goto command if you use:

"Type: Expression"

#### GoTo

Use the goto action to make a movie's timeline jump to a specific frame number, frame label, or scene, where it can stop or play from that point forward (depending on how you set it up).

Parameters for goto action:

Scene. Allows you to choose a scene as a starting point for the goto action. Once you've defined a scene, you can then define a frame number or label within that scene. When using this action within symbols, the scene parameter is not available. The options include:

<current scene>. Allows you to choose a frame number or label from the current scene as the point on the timeline to go to.

<next scene>. Causes the action to go to Frame 1 of the next scene. You cannot use this option to go to a specific frame number or label in the scene; for this you would use Scene\_Name.

Scene\_Name. Select a scene name from the list that appears.

Type. Based on the scene option you selected, this allows you to choose a specific frame in the scene to go to. The available options include the following:

Frame Number. Select a frame number to go to.

Frame Label. Select a frame label to go to.

Expression. Allows you to dynamically set the destination frame based on the value an expression evaluates to.

Next Frame. Causes the timeline to jump to the frame following the frame in which the action is triggered (this is only available when selecting <current scene> for the scene parameter.

Previous Frame. Causes the timeline to jump to the frame that preceded the frame in which the action was triggered.

Frame. If you chose Frame Number from the Type drop-down box, you would enter the frame number here. If you choose Frame Label, you enter the appropriate label. If you choose Expression, you enter the expression.

Go To and Play. Once the timeline has jumped to a specific frame, this option allows you to specify whether the movie will stop playing at the frame or continue playing from the at frame.

Note. The Go To action is Flash's way of creating hyperlinks within a Flash movie.

#### Play

Causes a movie to begin playing from its current position. If your movie stops due to a stop action, it cannot begin playing again until you use the Play action to start it. Play has no parameters. Use the Play action to create Start/Stop buttons for various timelines.

#### Stop

Will cause a movie to cease playing. You can use it during any points in the movie that you wish to remain visible for an extended period of time. Stop has no parameters.

#### Toggle High Quality

Turns antialiasing off and on, which affects the visual quality and playback speed of your movie. With antialiasing on, visual quality improves but playback slows on older computers. This action affects the entire movie.

Use this action to determine view quality and to turn off antialiasing for intensely animated portions of a movie.

#### Stop All Sounds

Halts all currently playing audio tracks. This action has no parameters.

#### Get URL

The Get URL action does two things: It either loads a specified URL into a browser window (in the same way an HTML hyperlink does), or it sends variable data in the movie to the specified URL. For example, variable data (such as that entered into a Flash form) can be sent to a CGI script

for processing in the same way an HTML form can. When using this action to send variables, only variables for the movie specified in the action are sent. The specified movie can be the main movie, a movie clip or a .swf file loaded using the Load Movie action. Although you will use the Get URL action primarily when your Flash movie exists on a Web page, you can also use it in a Flash projector to automatically open a browser window and display a specified URL.

#### Parameters:

**URL.** This is where you define the url for the action, either a relative path such as contents/movie1.html or an absolute path, such as http://www.mydomain.com/page1.html.

If your movie is on an HTML page, you can use this area to define a JavaScript function to call - such as javascript:newWindow() -when an event is triggered.

**Window.** Specifies the browser window or HTML frame in which to load and display the specified URL. If you have defined an HTML window or frames with a name and you want the specified URL to load into that window, simply type its name into this box. Other wise choose from:

**\_self.** Loads the specified URL into the window or frame now occupied by the Flash movie.

**\_blank.** Opens a new browser window and loads the specified URL into it.

**\_parent.** Opens the URL in the parent window of the current window.

**\_top.** If the Flash movie is in an HTML frame, this removes the frame set and loads the URL into the browser window.

**Variables.** Variable values in a movie can be sent to a server for processing. This option lets you choose how variables are dealt with when using the Get URL action. The following options are available.

**Don't send.** Doesn't send variables, best for simply opening a URL.

**Send using GET.** Sends variable values that are appended to the specified URL for processing by the server.

**Send using POST.** Sends variables separate from the URL, which makes it possible for you to send more variables. On regular HTML pages, this method is most frequently used to post information collected from a form to a CGI script on the server.

#### FS Command

FS command allows your Flash movie to communicate with other programs - say a Web browser or any program that can host your Flash movie (that is one in which you can embed your Flash movie).

Most people use this command to make their Flash movies interact with JavaScript on HTML pages and to control a Flash projector.

To use FS Command to open custom alert boxes:

1. Create a Flash movie with a button that includes a mouse event that triggers an FS Command action.
2. When setting up the FS Command, type Infobox in the Command box and "You will do everything I say!" in the Arguments box. You make up the command name, it can be anything you want, it's just a unique identifier.
3. If you wish to create a second button with an FS Command action on it, simply type Infobox in the Command box but "You will not listen to me" in the Argument box. You now have two buttons that use the same command but different arguments. You can now place your movie on the HTML page containing a JavaScript function that can detect when an FS Command has been activated in your movie.

**Command.** This is a unique name you assign to an FS Command; it can be anything you'd like.

**Arguments.** If the command requires that any information be passed - for example, to a JavaScript function - you must enter it here.

**Commands for stand-alone player.** You can use the following settings to control a Flash projector when distributing your movie as a stand-alone application:

**Fullscreen.** To display a projector so that it can be seen at full screen, choose true.

**Allowscale.** Allows the user to resize the Projector window.

**Showmenu.** Choose true to make the Projector menu available when the user right-clicks or Control clicks the projector window.

**Quit.** Quits the projector and closes its window.

**Exec.** Use this command to start an external application from Flash. Enter the directory path to the application in the Argument box.

#### Load Movie

\* Load a new movie into the Flash movie window or replace an existing one (which means you can display a new movie without loading a different HTML page).

\* Load an additional movie into the Flash movie window.

\* Load a movie into a movie clip target (thereby replacing the movie clip instance with an entire movie).

Parameters

URL. This is the directory path to the .swf file you want to load. It can be a relative path, such as mymovie.swf or an absolute path, such as http://www.mydomain.com/mymovie.swf

Location. This parameter defines the level or target that will be affected by the specified action.

Levels can be thought of as layers of separate .swf files stacked on top of each other in the Flash Player window. The number assigned to a level determines its position relative to the other levels. As the bottom .swf, file in the stack, Level 0 usually represents your original movie. The movie in Level0 sets the frame rate, background color and frame size for all other loaded movies. Movies can be loaded into levels that already contain another movie. Doing so simply replaces the existing .swf file on that level.

Target allows you to load an entire .swf into a space currently occupied by a movie-clip instance. Doing so causes the loaded .swf to inherit all of the movie clip's current properties, including its name, target path, size and position. In addition, if the target movie-clip instance is used in a motion tween or frame-by-frame animation, the loaded .swf will be used in its place.

Variables. If you want to pass the current movie's variables into the one that's being loaded, you use the parameter to determine how those variables are sent. This means that a movie can receive variable data before it's loaded so that it can react to that data immediately. The following options are available:

Don't send. Doesn't send variables.

Send using GET. Sends variables appended to the specified URL.

Send using POST. Sends variables separate from the URL.

Unload Movie

The Unload Movie action allows you to unload a movie that had been loaded into the movie window using the Load Movie action.

Parameters

Location defines the level or target containing the movie that will be unloaded.

Tell Target (Depreciated)

The Flash movie window can contain any number of timelines (the main movie, movie clips, and loaded movies). You use the Tell Target action -- which is always employed in conjunction with an action - to control one timeline from within another. A target is identified by either its level (if it was loaded into the movie window) or its instance name (if it is a movie clip). The main movie is identified as \_level0 or as \_root (/).

You can use the Tell Target action to start, stop, drag, hide, rotate, scale, and otherwise control any movie in the movie window.

Parameters

This command has only one parameter - target. This is where you define the movie that will be targeted for any actions.

If Frame is Loaded

If Frame Is Loaded is another command used to preface an action (and it's always used in conjunction with an action). The underlying logic goes something like this: If frame x is loaded, do these action. If frame x is not loaded, ignore the "If Frame is Loaded"

Command. This is known as a conditional statement: The action is only performed if the condition is met.

This command is commonly used to create a loop that constantly checks to see if the back end of a movie has been completely loaded, to create a preloader.

Instead of using Tell Target, FlashMX offers a "gotoAndPlay" or "gotoAndStop" commands with implicate paths. Find these actions hidden deeply in the Actions Tool Box:

Objects > Movie > Methods > gotoAndPlay In the object field you write your path to the targeted instance, it must be a Movie Clip instance that has been named. In the Parameters field you type in the frame to go to, either a number, this would be an expression, or a label that is a literal string and must be in quotes.