Shockwave and NetLingo

- Using ShockWave involves:
  - The ShockWave browser Plug-In that provides the runtime engine to execute your Director movie (this is analogous to the code added when you make a Projector)
  - A Director movie that you can "Shock" (prepare for the Web)
  - An HTML file to display the Shocked movie in your Browser

- Assign a Preferred Browser to Director (under **File →Preferences →Network**).

- Shocked movies are compressed by as much as 60 percent to reduce download time, and are decompressed by the Plug-In as they play. *Despite the compression, it is important to keep your file size to a minimum.* A good place to start is by reducing your Stage size - if your Director movie is 640x480, for example, you can shrink it down to 512x385 or 500x375. And be sure you're using 8-bit (or lower) graphics!

- Some operations are restricted from use in Shocked movies, notably those than read or write to/from the user's hard drive. XTRAs can only be used if they are already on the user's system, or are independently downloaded ahead of time by the user. You can set options that provide for automatic (prompted) downloading of XTRAs at runtime. You can't use Movies In A Window (MIAWs), although you can chain from one movie to another. The setPref/getPref functions offer some limited read/write capability on the user's drive – intended for things like preferences and "cookie"-like information.

- Your browser must be configured to recognize the Shockwave MIME type, which is "application/x-director", for file types ".dir", ".dxr", ".dcr".

- If you select the proper options under the "Formats" tab in the "Publish Settings" dialog under the "File" menu, Director will generate the complete HTML for you. You can either use this file as-is, or cut and paste it into another HTML document.

- Here's the HTML for Shockwave using the **<EMBED>** tag (height and width are your Stage size from **Modify →Movie →Properties** dialog)
  [Note: items in **bold** are to be entered exactly, non-bold items are specified by you]:

```
<EMBED SRC="mymovie.dcr" WIDTH="www" HEIGHT="hhh">
```
or
```
<EMBED SRC="mymovie.dcr" WIDTH="www" HEIGHT="hhh" NAME="MyMovie">
```

The "NAME" parameter allows JavaScript to reference your Shockwave applet.

- You can also code it so the Browser will automatically go to Macromedia's Shockwave Plug-In download page if the user doesn't have it installed:

```
<EMBED SRC="mymovie.dcr" WIDTH="www" HEIGHT="hhh"
      PLUGINSPAGE= "http://www.macromedia.com/shockwave/download/">
```

- More advanced usage with **<OBJECT>** tag. This will automatically download the Shockwave Plug-In if the user doesn't have it installed, or if a version earlier than 7.0.2.0 is installed:

```
<OBJECT  CLASSID="clsid:166B1BCA-3F9C-11CF-8075-444553540000"
CODEBASE="http://download.macromedia.com/pub/shockwave/cabs/director/sw.cab#version=7,0,2,0"
WIDTH="www" HEIGHT="hhh" NAME="sw" ID="sw">
<PARAM NAME="src" VALUE="mymovie.dcr">
<EMBED SRC="mymovie.dcr" WIDTH="www" HEIGHT="hhh"
      PLUGINSPAGE="http://www.macromedia.com/shockwave/download/">
</EMBED>
</OBJECT>
```

The "#version" part of the **CODEBASE** parameter specifies the version of Shockwave used to create your movie. It may change as newer versions of Shockwave are released. Older examples are: "#version=5,0,1,61" and "#version=6,0,0,0".

• Other parameters that can be used with **<EMBED>** and **<OBJECT>:**

"**TEXTFOCUS**" - value can be "**never**", "**onstart**", "**mouseUp**" . Determines when keyboard input to text fields starts being accepted by the Shockwave movie rather than the browser. It's still a good idea to include a "Click here to start" button to be safe. Example:

<EMBED SRC="mymovie.dcr**" WIDTH="**www**" HEIGHT="**hhh**" TEXTFOCUS="**onstart**">**

"**PALETTE**" - value can be "**foreground**" or "**background**". If set to "**foreground**", the palette is set to match the one specified in the Director file; "**background**" lets the browser assert the palette. Although you can force the palette in Director and use "**foreground**", it's generally a good idea to use a "browser-safe" palette.

(NOTE: Director 7 has removed support for the **EMBED** tag option "**PALETTE=Foreground**". Older movies that used this option will now be remapped to the web-safe palette by their browser. This may require re-authoring of some older Shockwave movies.) Example:

<EMBED SRC="mymovie.dcr**" WIDTH="**www**" HEIGHT="**hhh**" PALETTE="foreground">**

• Other HTML Parameters may be specified that you can read from within your Director movie. Some are pre-defined, or you can make custom ones Example:

<EMBED SRC="mymovie.dcr**" WIDTH="**www**" HEIGHT="**hhh**" MODE="active">**

Then, in Lingo:

```
set gPlayMode = externalParamValue("MODE")
```

The following Microsoft Internet Explorer can be used to pass info into a Shockwave movie:

swURL	A page, movie or Shockwave audio location
swText	Variable text used in fields
swForeColor	The foreground color of a sprite
swBackColor	The background color of a sprite
swFrame	A starting frame in the movie, or a target frame in the browser
swColor	A color value in the movie
swName	More text to be displayed in the movie, meant for username
swPassword	Passwords incorporated into your movies
swBanner	Variable text used in banner fields
swSound	The name of a sound cast member, or an indicator of whether sound is on or off
swVolume	A volume setting for the movie to use
swPreloadTime	A Shockwave audio setting
swAudio	The URL of a Shockwave audio file
swList	A comma-delimited list of anything
sw1 - sw9	Additional non-specific parameters

In HTML:

<PARAM NAME="swAudio" VALUE="mytune.swa**">**
<PARAM NAME="sw1" VALUE="active">

Then, in Lingo:

```
set gShockAudioURL = externalParamValue ("swAudio")
set gMyOwnParam = externalParamValue ("sw1")
```

• The following are some important considerations when authoring for Shockwave streaming:

- You must "Save and Compact" movies in order for them to stream properly.

- Casts whose Cast Properties are set to preload "Before Frame One" or "After Frame One" may force the bulk of a movie’s media to download before it begins to play (in authoring and projectors, but not Shockwave). To avoid this potential conflict altogether, ensure that the Cast is set to preload "As Needed" (the default).

- Compressed fonts preload their media prior to movie playing, significantly increasing the time it takes for the first frame to appear on playback.

NetLingo

• NetLingo describes a set of commands and functions that extend Lingo's functionality to Web-specific, or network, operations. Note that certain operations happen asynchronously. That is, you issue a command to start an operation, then check the status to check when and if the operation completes. This may involve the use of a network ID, returned by certain NetLingo functions and used to test if that particular operation is complete.

gotoNetMovie

Launches the specified Shockwave movie, either relative or using a complete pathname:

```
gotoNetMovie ("http://www.mysite.com/mymovie.dcr")
gotoNetMovie ("nextmovie.dcr")
set myNetID = gotoNetMovie
    ("http://www.mysite.com/mymovie.dcr#Contents")
```

The last form does two things – goes to the frame specified by the marker label "Contents" in the movie, and it also returns a network ID that can be tested later.

If a gotoNetMovie operation is in progress, and you issue a second gotoNetMovie command before the first is finished, the second command cancels the first.

It is good practice to loop playback after executing a gotoNetMovie to ensure that the current movie is still running when enough of the calling movie is ready to be played. For example, inserting a "go the frame" script on a frame after the execution of the gotoNetMovie would ensure the initial movie is running, and therefore a successful execution of the gotoNetMovie.

gotoNetPage

Launches a new HTML page:

```
gotoNetPage ("http://www.anothersite.com/home.html")
gotoNetPage ("newpage.html")
gotoNetPage ("newpage.html", "targetwindow")
```

If targetName is specified, and is a window or frame in the browser, gotoNetPage replaces the contents of that window or frame. If targetName isn't a frame or window that is currently open, goToNetPage opens a new window.

downloadNetThing URL, localPath

Downloads the file specified by URL into the local path specified by *localPath*. Any kind of file may be downloaded this way. The file is copied while the current movie continues playing. Use **netDone()** to find out whether downloading is finished. This is useable in web-enabled Projectors running locally, but not from Shocked movies. This protects users from unintentionally copying files from the Internet.

Although many network operations can be active at one time, running more than four concurrent operations usually slows down performance unacceptably. Neither the Director movie's cache size nor the setting for the Check Documents option affects the behavior of the downloadNetThing command.

Director for Java does not support downloadNetThing.

Example: downloads an external cast from a URL to the Director application folder, and then points the cast library called "Webcast" to that external cast file:

```
downLoadNetThing("http://www.myURL.com/newcast.cst", ¬
    (the moviePath&"mywebcast.cst") )
set the fileName of castLib "Webcast" = ¬
    (the moviePath&"mywebcast.cst")
```

preLoadNetThing URL

Pre-downloads the item specified by URL into the browser's cache so it will be ready to play when needed.

You can capture the returned network ID to use if you wish to monitor the progress of the operation with netDone() or netError().

```
preLoadNetThing ("http://www.myURL.com/nextmovie.dir")
```

Director player for Java doesn't support **preLoadNetThing** because Java's security model doesn't allow writing to the local disk.

The **preLoadNetThing** function downloads the file while the current movie continues playing. Use **netDone()** to find out whether downloading is finished.

Although many network operations can be active at a time, running more than four concurrent operations usually slows down performance unacceptably. Neither the cache size nor the Check Documents option in a browsers Preferences affects the behavior of the command.

externalParamCount()

Returns the number of external parameters in the HTML page. This function is valid only for Shockwave movies that are running in a browser. It doesn't work for movies during authoring or in projectors.

externalParamValue

Returns the value of a parameter specified in the HTML page.
The parameter name or number in sequence is specified:

```
set var = externalParamValue ("myparam")
set anothervar = externalParamValue (2)
```

Note that the match for a specified parameter name is not case-sensitive. If no matching parameter exists, returns <Void>.

externalParamName

Returns the name of a parameter specified in the HTML page.
The parameter name or number in sequence is specified:

```
set var = externalParamName ("myparam")
set thename = externalParamName(2)
```

Note that the match for a specified parameter name is not case-sensitive. If no matching parameter exists, returns <Void>.

setPref (prefFileName, prefValue)

Allows you to save preferences-type info on the user's hard drive in a safe way (akin to "cookies"):

```
setPref ("my9876.txt", "beginner")
```

The file is a standard text file, and prefFileName must be a valid file name. Pick a unique filename, and conform to "8.3" naming conventions to assure multi-platform compatability.

If the movie is playing in a browser, a folder named "Prefs" is created in the "Plug-In Support" folder. The setPref command can write only to that folder.

If the movie is playing in a projector or within Director, a folder is created in the same folder as the application. The folder has the name "Application Folder", where "Application" is the name of the Director movie or projector. For example, a projector named "BigBand" would have a folder named "BigBand Folder".

getPref (prefFileName)

Retrieves info from a file previously stored with setPref():

```
set prefText = getPref ("my9876.txt")
```

If no such file exists, getPref returns <Void>. The file name used for prefFileName must be a valid file name only, not a full path – Director supplies the path. The only valid file extensions for prefFileName are .txt and .htm – any other extension is rejected.

Do not use this command to access read-only or locked media.

Important Note: In a browser, data written by setPref is not private. Any Shockwave movie can read this information and upload it to a server. Confidential information should not be stored using setPref.

getNetText

Retrieves text from a file, usually on a server, or initiates a CGI query. Useful for setting variables, lists, etc. inside your Director movie:

```
    getNetText ("infofile.txt")
    getNetText ("http://www.myserver.com/infofile.txt")
```

Once getNetText is called, you must wait for completion (using netDone), and then read the actual text using netTextResult().

Here's an example which submits a CGI query:

```
    getNetText("http://www.myserver.com/¬
        cgi-bin/query.cgi?name=Sally%20Smith")
```

This is the same as the previous example, but it uses a property list to submit the CGI query, and does the URL-encoding for you to make things a little easier to read and manipulate in Director:

```
    getNetText("http://www.myserver.com/¬
        cgi-bin/query.cgi", [#name="Sally Smith"])
```

For a movie that plays back as a Java applet, getNetText retrieves text only from the same domain as the applet. This differs from Shockwave and is necessary due to Java's security model.

netTextResult()

Returns the result of the preceding getNetText call. Before calling, you should make sure the operation completed, using netDone().

netDone()

Returns TRUE when the previously requested network has completed. **NOTE: Do not use this in a repeat loop, or your movie may appear to be hung. Instead, check using a frame loop**.

URLEncode (proplist_or_string, serverOSString, characterSet)

Returns the URL-encoded string for its first argument. Allows more flexible specification of CGI parameters. The same translation is done as for getNetText when it is given a property list. (See Director help for explanations of the optional parameters serverOSString and characterSet(. Example: URLEncode supplies the URL-encoded string to a CGI query at the specified location:

```
    URL = "http://myserver/cgi-bin/echoquery.cgi" ¬
            gotoNetPage (URL & "?" & ¬
                URLEncode( [#name: "Ken", #hobby: "What?"] )
```

getLatestNetID()

Returns a unique ID number for a preceding net request, such as getNetText(). This number may then be used by netDone() and netTextResult() to work with results of a specific net request:

<u>In Lingo, with no network ID:</u>

```
on getText
    getNetText ("infofile.txt")
    go frame "NetWait"
end
```

Then, in the frame "NetWait":

```
on exitFrame
    global gMyText

    if netDone() then
        set gMyText = netTextResult()
        go frame "Done"
    else
        go the frame
    end if
end
```

<u>Using getLatestNetID:</u>

```
on getText fileName
    global gTextStreamID

    getNetText (fileName)
    set gTextStreamID = getLatestNetID()
end
```

Then:

```
if netDone (gTextStreamID)
    set gMyText = netTextResult(gTextStreamID)
end if
```

netError()

Returns an empty string until the specified network operation is complete. Once the operation is complete, returns "OK" if the operation was successful, an error string if not (returns 0 or an error code when running in Director or as a projector). Use without parameters to test the last network operation, or with a specified net ID to test a particular network operation:

```
if netError() then …
if netError (myNetID) then …
```

The string's content comes from Java and can vary on different operating systems or browsers. The text may not be translated into the local language.

When operating on an invalid URL, it is still conceivable to have netError() return "OK." It is common for servers to return error pages when a requested URL does not exist. These error pages usually have a mime-type of "text/html" and are therefore reported as successful transfers by netError(). To verify a successful operation, it is recommended that you test for the expected mime-type with "netMime()" in addition to netError().

netMIME()

Returns the MIME type of the Internet file that the last network operation returned (the most recently downloaded HTTP or FTP item). The function can be called only after netDone() and netError() report that the operation is complete and successful. After the next operation starts, the Director movie or projector discards the results of the previous operation to conserve memory. Example:

```
-- Loop on the current frame until we get a recognized object:

on checkNetOperation theURL
    if netDone (theURL) then
        set myMimeType = netMIME()

        case myMimeType of
            "image/jpeg":
                go frame "jpeg info"
            "image/gif":
                go frame "gif info"
            "application/x-director":
                goToNetMovie theURL
            "text/html":
                goToNetPagae theURL
            otherwise:
                alert "Please choose a different item."
        end case
    else
        go the frame
    end if
end
```

netAbort()	Will try to halt a network operation. A net ID should be given. If not available, then the URL should be specified. The URL must be identical to that used to begin the network operation.
netLastModDate()	Returns a string containing the modification date of an item downloaded with getNetText() or preLoadNetThing().The netLastModDate function can be called only after netDone() and netError() report that the operation is complete and successful. After the next operation starts, the Director movie or projector discards the results of the previous operation to conserve memory.
the netPresent	Returns TRUE if the Net Support XTRAs are available. Does not check for an actual network connection. (May also be called as a function: netPresent() – but this form will cause a script error if the needed XTRAs are not available.)
netStatus "message"	Displays the specified message in the browser's status bar area. In authoring mode, the string is displayed in the Message window. Does nothing in a projector. Not supported in all browsers.
frameReady	Reports if the members in a specified frame or range of frames has been downloaded from the Server. Returns TRUE when ready:
	`if frameReady (5) then ...              --frame 5`
	`if frameReady (5, 20) then ...          --frames 5 through 20`
	`if frameReady (sprite 12, 5) then …  --sprite12 in frame 5`
mediaReady	Tests to see if a specific cast members has been downloaded the Server. Can be used in member or sprite form. Returns TRUE when ready:
	`if the mediaReady of member "background PICT"then ...`
	`if the mediaReady of sprite 15 then ...`
tellStreamStatus	When set to TRUE, this turns on the feature that reports the status of every network action to the on streamStatus handler
	`tellStreamStatus(TRUE)        --turn it on`
	`if tellStreamStatus() then …`

on streamStatus URL, state, bytesSoFar, bytesTotal, error

When turned on by tellStreamStatus, this is called periodically by all network operations. Director automatically fills in the parameters with information regarding the progress of downloads. These parameters can be examined to keep track of the status of network operations.
URL = the Internet address of the data being retrieved.
state = the state of the stream being downloaded. Possible values are "Connecting", "Started", "InProgress", "Complete", and "Error".
bytesSoFar = the # of bytes retrieved from the network so far.
bytesTotal = the total number of bytes in the stream, if known. The value may be 0 if the HTTP server does not include the content length in the MIME header.
error = string containing "" (EMPTY) if the download has not completed, "OK" if it completed successfully, or an error code if it failed.

The handler is called by Director automatically, and there is no way to control when the next call will be. If information regarding a particular operation is needed, use getStreamStatus.

getStreamStatus (netID)
getStreamStatus (URLString)

Returns a property list matching the format used for the globally available tellStreamStatus function. The contents of the list are:
#URL = string containing the URL location used to start the network operation.
#state = string consisting are "Connecting", "Started", "InProgress", "Complete", "Error", or "NoInformation". (this last string is for the condition when either the netID is so old that the status information has been dropped, or the URL specified was not found in cache).
#bytesSoFar = # of bytes retrieved from the network so far.
#bytesTotal = total number of bytes in the stream, if known. The value may be 0 if the HTTP server does not include the content length in the MIME header.
#error = string containing "" (EMPTY) if the download is not complete, "OK" if it completed successfully, or an error code if it ended with an error.

Example: Displays in the message window the current status of a download begun with getNetText() and the resulting net ID stored in the variable netID:

```
put getStreamStatus(netID)
-- [#URL: "www.macromedia.com", #state:"InProgress",  ¬
-- #bytesSoFar: 250, #bytesTotal: 50000, #error: EMPTY]
```

the URL of member member
the streamName of member member

Two forms to specify the URL location of Shockwave Audio (SWA) and Flash movie cast members. The second form works only for SWA members.

For Flash movie members, the URL property is synonymous with the pathName member property.

This property can be tested and set. For SWA members, it can be set only when the SWA streaming cast member is stopped.

```
set the URL of member "Benny Goodman" = ¬
    "http://audio.macromedia.com/samples/classic.swa"
```

browserName()

When called with no parameters, reports the name of the currently specified browser. You can also set it to a full pathname to change the browser (note the parentheses!):

```
set currentBrowser = browserName()
browserName ("myDrive:Netscape Communicator™ 4.7")
```

The form browserName(#enabled, trueOrFalse) determines whether the specified browser launches automatically when the goToNetPage command is issued.

NOTE: This command is only useful playing back in a projector or Director, and has no effect when playing back in a browser.

cacheSize()

When called with no parameters, returns the current cache size for Web-enabled projectors. You can also set it to adjust the size. This function is valid only for movies running in Director or as projectors. Note that this is Director's cache size, not the browser's—this function is not valid for Shockwave movies because they use the settings of the browser in which they run. Example:

```
on checkCache
    if cacheSize()<1000 then
        alert "increasing cache to 1Mb"
        cacheSize (1000)
    end if
end checkCache
```

the environment.internetConnected

The system property "the environment" contains a property list with several pieces of information about the current Director environment. The #internetConnected property indicates whether there is an active internet connection:

- #online -- there is an active Internet connection
- #offline -- there is not an active Internet connection
- #unknown -- Internet connection status is not known

externalEvent "string"

Sends a string to the browser that the browser can interpret as a scripting language instruction, allowing a movie playing or a browser to communicate with the HTML page in which it is embedded. The string sent by externalEvent must be in a scripting language supported by the browser. This works only for movies in browsers. To enable communication between an applet and a browser, use Java, JavaScript, or VBScript.

The command does not return a value. There is no immediate way to determine if the browser handled the event or ignored it. Use "EvalScript" within the browser to return a message to the movie, which would catch it using "on EvalScript".

Example 1: these statements use externalEvent in the LiveConnect scripting environment, which is supported by Netscape 3.x and later. LiveConnect evaluates the string passed by externalEvent as a function call. JavaScript authors must define and name this function in the HTML header. In the Director movie, the function name and parameters are given as a string to the externalEvent call. Because the parameters must be interpreted by the browser as separate strings, each parameter is surrounded by single quotation marks:

In HTML:

```
function MyFunction(param1, param2) {
    //script here
}
```

In Director:

```
externalEvent ( "MyFunction('param1','param2')" )
```

Example 2: These statements use externalEvent in the ActiveX scripting environment used by Internet Explorer on Windows. ActiveX treats externalEvent as an event and processes this event and its string parameter the same as an onClick event in a button object.

In HTML:
Define a function in the HTML header to catch the event. Here's an example in VBScript:

```
        Sub
        NameOfShockwaveInstance_externalEvent(aParam)
            'script here
        End Sub
```

Alternatively, define a script for the event:

```
        <SCRIPT FOR="NameOfShockwaveInstance"
        EVENT="externalEvent(aParam)"
        LANGUAGE="VBScript">
            'script here
        </SCRIPT>
```

In Director:
Include the function and any parameters as part of the string for externalEvent:

```
      externalEvent ("MyFunction ('parm1','parm2')" )
```