



# WebXtra<sup>®</sup> Help

[Installation](#)

[Getting Started](#)

[Properties](#)

[Functions](#)

[Events](#)

[Creating Projectors](#)

[How to Order & Register](#)

[Licensing & Availability](#)

[Technical Support](#)

For up-to-date information please visit our web site:  
[xtras.tabuleiro.com](http://xtras.tabuleiro.com)



## **WEBXTRA HELP: INSTALLATION**

The installation procedure is slightly different depending on the version of Director and platform used, and the location where Director is installed on your system. Make sure you have administrative rights to create files in the directory where Director is installed on your system.

**Important:** please remember that WebXtra sprites do not spawn a working web browser on the Macintosh: the Xtra is only fully functional on Windows. The Macintosh versions of the Xtra are provided as stubs to help users that must author cross-platform movies on the Mac platform, or need to work in both platforms with the same set of files and cast libraries, without any error messages. With these stub Xtras it is perfectly possible to write a complete application using WebXtra on the Mac and cross publish it for Windows using Director MX 2004.

### **INSTALLING THE XTRA ON WINDOWS - Director 8.5 and Director MX**

By default the installer will unpack the Xtra, documentation and sample files to the directory C:\WebXtra\_4. To install the Xtra, just copy the file WebXtra.x32 to the Director 8.5 or Director MX XTRAS folder. If you have previously installed an older copy of the Xtra make sure to remove or replace it.

These are the final pathnames for the Xtra file after being copied to the default location of the Xtras folder in each application:

**Director 8.5- C:\Program Files\Macromedia\Director 8.5  
\Xtras\WebXtra.x32**

**Director MX- C:\Program Files\Macromedia\Director  
MX\Xtras\WebXtra.x32**

No additional configuration is needed. Restart Director for the changes to take effect. The Xtra should appear in the INSERT->TABULEIRO XTRAS->WebXtra menu item.

### **INSTALLING THE XTRA ON WINDOWS - Director MX 2004**

By default the installer will unpack the Xtra, documentation and sample files to the directory C:\WebXtra\_4. To install the Xtra, just copy the file WebXtra.x32 to the Director MX 2004 XTRAS folder. If your copy of Director MX 2004 is installed at the default location, the Windows Xtra file will be located at:

**C:\Program Files\Macromedia\Director MX 2004  
\Configuration\Xtras\WebXtra.x32**

Now you need to install the files necessary for creation of cross-platform projector for Mac OSX. Go back to the directory where the Xtra files were unpacked, by default at C:\WebXtra\_4. Open the "Cross Platform Resources" directory. Now copy the files "WebXtra.data" and "WebXtra.rsrc" files to the "Configuration\Cross Platform Resources\Macintosh\Xtras" directory used by Director MX 2004. In a default installation of Director these files will end up at the following locations:

**C:\Program Files\Macromedia\Director MX 2004  
\Configuration\Cross Platform**

**Resources\Macintosh\Xtras\WebXtra.data  
C:\Program Files\Macromedia\Director MX 2004  
\Configuration\Cross Platform  
Resources\Macintosh\Xtras\WebXtra.rsrc**

Finally, you need to edit the xtrainfo.txt file to include information about WebXtra. This information is used by the cross-platform publishing features in Director MX 2004, to locate the files needed when assembling the OSX version of your projector. The xtrainfo.txt file is located by default at:

**C:\Program Files\Macromedia\Director MX 2004  
\Configuration\xtrainfo.txt**

Double click the file to open it in notepad, or alternatively edit with any other text editor. You need to add the following line to the end of the file:

**[#namePPC:"WebXtra", #nameW32:"WebXtra.x32"]**

Restart Director for the changes to take effect. The Xtra should appear in the INSERT->TABULEIRO XTRAS->WebXtra menu item.

### **INSTALLING THE XTRA ON MAC OS 8 AND 9 - Director 8.5**

The installer will unpack the Xtras, documentation and sample files to a folder named "WebXtra 4 " on your machine. To install the Xtra just copy the file "WebXtra" from the MacClassic folder to the Xtras folder of your Director 8.5 installation. The final pathname for the Xtra will be for example:

**Macintosh HD:Applications:Macromedia Director  
8.5:Xtras:WebXtra**

No additional configuration is needed. Restart Director for the changes to take effect. The Xtra should appear in the INSERT->TABULEIRO XTRAS->WebXtra menu item in Director. Please remember that WebXtra sprites do not spawn a working browser on the Mac: the Xtra is provided only as a convenience for authors working on cross-platform movies. The stub Xtra recognizes all standard WebXtra functions and properties, and can be used to avoid error messages when opening cast libraries that contain WebXtra castmembers.

### **INSTALLING THE XTRA ON MAC OSX 10 - Director MX**

The installer will unpack the Xtras, documentation and sample files to a folder named "WebXtra 4 " on your machine.

The first step is to copy the OSX version of the Xtra, which will be used in the authoring environment and also when creating OSX projectors. This file is located in the install disk image, at:

**WebXtra 4 Folder/MacOSX/WebXtra**

This file needs to be copied to the Director MX Xtras folder. The final pathname for the OSX Xtra in a default installation of Director MX will be:

**OSX Volume Name/Applications/Macromedia Director  
MX/Xtras/WebXtra**

Director MX running on Mac OSX can also be used to create Classic projectors, for Mac OS versions 8 and 9. In order to enable this feature you

need to copy the Classic version of WebXtra to the correct location in your Director MX installation. First locate the Classic version of WebXtra in the install disk:

**WebXtra 4 Folder/MacClassic/WebXtra**

This file needs to be copied to the following location in the Director MX folder:

**OSX Volume Name/Applications/Macromedia Director MX/Classic MacOS/Xtras/WebXtra**

This will allow creation of both OSX and Classic Projectors from Director MX, with the right version of the stub Xtra being bundled automatically with your Projector. Restart Director for the changes to take effect. The Xtra should appear in the INSERT->TABULEIRO XTRAS->WebXtra menu item the next time Director MX is started. Please remember that WebXtra sprites do not spawn a working browser on the Mac: the Xtra is provided only as a convenience for authors working on cross-platform movies. The stub Xtra recognizes all standard WebXtra functions and properties, and can be used to avoid error messages when opening cast libraries that contain WebXtra castmembers.

**INSTALLING THE XTRA ON MAC OSX 10 - Director MX 2004**

The installer will unpack the Xtras, documentation and sample files to a folder named "WebXtra 4 " on your machine.

The first step is to copy the OSX version of the Xtra, which will be used in the authoring environment and also when creating OSX projectors. This file is located in the install disk image, at:

**WebXtra 4 Folder/MacOSX/WebXtra**

This file needs to be copied to the Director MX Xtras folder. The final pathname for the OSX Xtra in a default installation of Director MX will be:

**OSX Volume Name/Applications/Macromedia Director MX 2004/Configuration/Xtras/WebXtra**

Director MX 2004 running on Mac OSX can also be used to create Classic projectors, for Mac OS versions 8 and 9. In order to enable this feature you need to copy the Classic version of WebXtra to the correct location in your Director MX installation. First locate the Classic version of WebXtra in the install disk:

**WebXtra 4 Folder/MacClassic/WebXtra**

This file needs to be copied to the following location in the Director MX folder, to be used for cross-platform publishing. Copy it to:

**OSX Volume Name/Applications/Macromedia Director MX 2004/Configuration/Cross Platform Resources/Classic MacOS/Xtras/WebXtra**

Fully functional Windows projectors containing WebXtra browsers can be created directly on Director MX 2004 running on Mac OSX, after installation of the Windows version of the Xtra. It is located on the install disk, at:

## **WebXtra 4 Folder/Windows/WebXtra.x32**

Copy this file to the Cross Platform resources directory in Director MX 2004, so that it will be available at:

**OSX Volume Name/Applications/Macromedia Director MX  
2004/Configuration/Cross Platform  
Resources/Windows/Xtras/WebXtra.x32**

Finally, you need to edit the xtrainfo.txt file to include information about WebXtra. This information is used by the cross-platform publishing features in Director MX 2004, to locate the files needed when assembling the Classic MacOS and Windows versions of your projector. The xtrainfo.txt file is located by default at:

**OSX Volume Name/Applications/Macromedia Director MX  
2004/Configuration/xtrainfo.txt**

Double click the file to open it in TextEdit, or alternatively edit with another text editor. Make sure to save the file in plain text format, though. You need to add the following line to the end of the file:

```
[#namePPC:"WebXtra", #nameW32:"WEBXTRA.X32"]
```

Restart Director for the changes to take effect. The Xtra should appear in the INSERT->TABULEIRO XTRAS->WebXtra menu item, and will also be available for publishing of cross-platform projectors. Please remember that WebXtra sprites do not spawn a working browser on Mac projectors or the authoring environment: the Xtra is provided only as a convenience for authors working on cross-platform movies. The stub Xtra recognizes all standard WebXtra functions and properties, and can be used to avoid error messages when opening cast libraries that contain WebXtra castmembers. Windows projectors cross-published in Director MX 2004 have no limitations.



## WEBXTRA HELP: GETTING STARTED

WebXtra is an Asset Xtra. Unlike scripting Xtras, Asset Xtras can be manipulated using the score and cast windows, and their properties can be adjusted through scripting, just like Director's built-in media types.

To create a WebXtra cast member, go to the INSERT menu, select TABULEIRO Xtras->WebXtra. If you are using an unregistered version of the Xtra, Director will display the WebXtra About Box, where you can enter your serial number and registration information. If the Xtra is already registered this dialog box will not be displayed.

A new cast member will appear in your Cast Window. It has the WebXtra icon and thumbnail. Click on it and drag the cast member to the score. A new sprite will be created on your score window, and it can be used to adjust the position and the size of the WebXtra browser window. You can now use scripting functions or the pre-made WebXtra behaviors available with the product to control the browser component.

For a simple test, please add the following Lingo behavior to the WebXtra sprite:

```
on beginsprite me
    sprite(me.spriteNum).Navigate("www.macromedia.com")
end

on exitframe
    go the frame
end
```

Users of Director MX 2004 can also use JavaScript syntax:

```
function beginsprite() {
    sprite(this.spriteNum).Navigate
    ("www.macromedia.com")
}

function exitframe() {
    movie.go( movie.frame)
}
```

Save the behavior, and play your Director movie. The WebXtra browser will initialize and display the Macromedia web site.



## **WEBXTRA HELP: PROPERTIES**

WebXtra cast members have only one property:

**browserServicesAvailable** - used to determine if a working browser sprite can be created. WebXtra can only create a working browser sprite when Internet Explorer 5, 5.5 or 6 is installed, and only in the Windows platform. This property will return FALSE or 0 when the Xtra is running on the Mac platform or in a Windows machine that does not provide the required services. In these cases the Xtra will still operate without any error messages, but will create just a white sprite instead of a fully operational browser window.

*Lingo example:*

```
if member(x).browserServicesAvailable = false then
    go to frame "nobrowseravailable"
end if
```

*JavaScript syntax example:*

```
if (member(x).browserServicesAvailable == false) {
    _movie.go("nobrowseravailable")
}
```

WebXtra sprites have several properties that can be used to obtain information from the browser object, or set its behavior. Most properties are available only after the browser is initialized, so make sure to check the information returned for VOID values.

**busy** - Read-only property, indicates if the browser is currently busy or conducting a network operation. It is generally used to display an animation indicating browser activity. Possible return values are TRUE or FALSE (1 or 0).

*Lingo example:*

```
put sprite(x).busy
-- 0
```

*JavaScript syntax example:*

```
if (sprite(x).busy) {
    trace("busy")
}
```

**offline** - Read-only property, indicates if the browser is currently operating in offline mode. It is generally used to detect if an internet connection is available. Possible return values are TRUE or FALSE (1 or 0).

*Lingo example:*

```
put sprite(x).offline
-- 0
```

*JavaScript syntax example:*

```
if (sprite(x).offline) {
    trace("no internet connection available")
}
```

**silent** - This property can be tested and set. When the browser is in silent mode it will not display dialog boxes with error messages and warnings to

the end user, for example when a page that contains invalid JavaScript code is loaded. Possible values are TRUE or FALSE (1 or 0).

*Example:*

```
sprite(x).silent = 1
```

**image** - Read-only property. Takes a screenshot of the browser window and returns an image object, ready to be used with imaging Lingo functions. Please make sure the browser window is not covered by other elements when this function executes.

*Lingo example:*

```
put sprite(x).image
```

```
--<image:202bc8>
```

*JavaScript syntax example:*

```
trace (sprite(x).image)
```

```
//<<image:202bc8>>
```

**html** - This property can be tested and set. It returns the current HTML code of the main page loaded in the browser window, and can also be used to pass an HTML string to be loaded and displayed by the browser. Important: getting the HTML property stops and cancels any browser operations currently in place, so you may want to test if the browser is busy before issuing this command.

*Lingo example:*

```
-- Get the HTML text currently loaded on the browser
```

```
member(x).text = sprite(x).html
```

```
-- Display new HTML content
```

```
sprite(x).html = "My HTML string here"
```

*JavaScript syntax example:*

```
trace (sprite(x).html)
```

```
//<BODY>my text here</BODY>
```

**title** - Read-only property. Returns the title of the current page loaded in the browser. Applications usually rely on the event "titleChange" to display this information without needing to poll the browser continuously.

*Lingo example:*

```
put sprite(x).title
```

```
--"Home Page"
```

*JavaScript syntax example:*

```
trace (sprite(x).title)
```

```
//"Home Page"
```

**url** - Read-only property. Returns the url of the current page loaded in the browser. Applications usually rely on the events "startNavigation" or "openNewWindow" to store this information without needing to poll the browser continuously.

*Lingo example:*

```
put sprite(x).url
```

```
--"http://xtras.tabuleiro.com"
```

*JavaScript syntax example:*

```
trace (sprite(x).url)
```

```
//"http://xtras.tabuleiro.com"
```

**browserReference** - Read-only property. Returns an unique browser

reference ID number that identifies this WebXtra sprite. This number can be used with the "openNewWindow" event to redirect creation of new windows to existing browser sprites. The "Complete Browser" sample included with the Xtra uses this property to redirect new windows requests to a browser sprite hosted in a MIAW, please examine the script "WebBrowser\_Behavior" for more information on how it is used.

*Lingo example:*

```
put sprite(x).browserReference  
--145628
```

*JavaScript syntax example:*

```
trace (sprite(x).browserReference)  
//76863
```



## **WEBXTRA HELP: FUNCTIONS**

WebXtra cast members have only one function:

**member(x).register(serialNumber)** - This function can be used in Projectors to register the Xtra at runtime, and allow developers to save linked casts. It is not necessary for the normal operation of the Xtra, as WebXtra castmembers already save their registration status in the Director movie.

*Example:*

```
member("mpegfile").register("SERIALNUMBERHERE")
```

WebXtra sprites have several properties that are used to invoke operations of the browser component. These functions are usually called by other interface elements on your movie, such as a menu item used to print the contents of the browser window or a button used to go back to the last page visited in the history.

**sprite(x).navigate(url)** - Used to initiate a navigation to the specified URL.

*Example:*

```
sprite(1).navigate("www.macromedia.com")
```

**sprite(x).navigateFrame(url, frameName)** - Used to initiate a navigation to the specified URL in a given HTML frame. This is useful if you want to have script controls in Director that change the content of just one frame while a frameset is loaded in the browser.

*Example:*

```
sprite(1).navigateFrame("https://www.secure.com",  
"_contentFrame")
```

**sprite(x).scrollWindow(xDelta, yDelta)** - This function can be used to create a button that scrolls the page loaded in the browser from Lingo, if your interface is not displaying the standard browser scroll bars. It accepts negative and positive values.

*Example:*

```
sprite(1).scrollWindow(-10, 100)
```

**sprite(x).goBack()** - Navigates to the previous URL in the browser history, if available.

*Example:*

```
sprite(1).goBack()
```

**sprite(x).goForward()** - Navigates to the next URL in the browser history, if available.

*Example:*

```
sprite(1).goBack()
```

**sprite(x).goHome()** - Navigates to the Home URL set in the system

preferences for the Internet Explorer browser.

*Example:*

```
sprite(1).goHome()
```

**sprite(x).goSearch()** - Navigates to the default SEARCH URL set in the system preferences for the Internet Explorer browser.

*Example:*

```
sprite(1).goSearch()
```

**sprite(x).stop()** - Stops the current browser operation, aborting any network transfer in progress.

*Example:*

```
sprite(1).stop()
```

**sprite(x).refresh()** - Reloads the current URL. Default cache settings will be used to speed up the loading if necessary.

*Example:*

```
sprite(1).refresh()
```

**sprite(x).refresh2()** - Reloads the current URL, but forces the browser to ignore any cached information.

*Example:*

```
sprite(1).refresh2()
```

**sprite(x).browserSaveAs()** - Opens a standard SAVE AS dialog that can be used to select a filename and options to save the HTML document currently loaded in the browser.

*Example:*

```
sprite(1).browserSaveAs()
```

**sprite(x).browserPageSetup()** - Opens the standard PAGE SETUP dialog that can be used to adjust page properties for future printing operations.

*Example:*

```
sprite(1).browserPageSetup()
```

**sprite(x).browserPrintPreview()** - Opens the standard PRINT PREVIEW window, where the user can visualize a print job before actually invoking the final print operation.

*Example:*

```
sprite(1).browserPrintPreview()
```

**sprite(x).browserPrint()** - Opens the standard PRINT dialog that can be used to select a target printer and print the current HTML page.

*Example:*

```
sprite(1).browserPrint()
```

**sprite(x).browserPrintNoPrompt()** - Prints the current HTML page without prompting the user for confirmation

*Example:*

```
sprite(1).browserPrintNoPrompt()
```



## **WEBXTRA HELP: EVENTS**

A new addition to WebXtra 4 is the concept of events. The Xtra sprite will generate scripting events that are sent directly to scripts attached to it, and pass the normal message chain in Director, reaching frame and movie scripts if they are not handled at the sprite level. This avoids the need to poll the browser for information continuously, as your scripts will be informed of any changes necessary. All events pass the "spriteRef" as the first parameter, and this value can be used to extract information about the sprite that generated the event directly.

Some events accept return values to indicate how the browser should react to a specific situation. Using events it is possible for example to block the creation of popup windows or authorize navigation to a specific link or URL, before it takes place. If you need more information please study the "WebBrowser\_Behavior" script, attached to the "Complete Browser" example that ships with the Xtra. This is a good example of a simple behavior showing how your scripts can intercept events and act on them.

It is not necessary to implement a script to handle all events if you are not planning to use them: the Xtra will assume default values if an event is not handled. You can implement handlers only for the events that are appropriate for your movie, or you can use the pre-made behaviors available at our site as a starting point for your own scripts. Below are the events generated by WebXtra 4:

**displayScrollBars(spriteRef)** - Generated by the sprite when a browser window is create or refreshed, to control the display of outer scroll bars in the main window frame. Scripts can return FALSE to prevent scroll bars from appearing. The default behavior of the browser is to automatically create scroll bars.

*Lingo example:*

```
on displayScrollBars spriteRef
    return false
end
```

*JavaScript syntax example:*

```
function displayScrollBars(spriteRef){
    return false
}
```

**showContextMenu(spriteRef)** - Called by the Xtra when the user clicks with the right mouse button over the HTML area, to invoke the standard browser context menu. You can return TRUE to allow the menu to appear (the default value), or FALSE to prevent the menu from being displayed.

*Lingo example:*

```
on showContextMenu spriteRef
    return false
end
```

*JavaScript syntax example:*

```
function showContextMenu(spriteRef){
    return false
}
```

**navigateError(spriteRef, URL, frameName)** - This event is generated

when the browser can not load an URL. It may be a malformed URL or one that is not currently online. The event parameters include the name of the URL that failed as well as the name of the frame containing it if the navigation was invoked in an HTML frame. The default action is to display the standard error URL for Internet Explorer, but you can instead return FALSE from this handler and present your own customized error message.

*Lingo example:*

```
on navigateError spriteRef, URL, frameName
    alert("There was an error trying to load "&URL)
    return false
end
```

*JavaScript syntax example:*

```
function navigateError(spriteRef, URL, frameName ){
    _player.alert("There was an error trying to load "+URL)
    return false
}
```

**fileDownload(spriteRef)** - Called by WebXtra when a local file is opened or a download operation is started. The operation can be canceled by returning FALSE. The default value is to let the download operation to start.

*Lingo example:*

```
on fileDownload spriteRef
    return false
end
```

*JavaScript syntax example:*

```
function fileDownload(spriteRef ){
    return false
}
```

**startNavigation(spriteRef, URL, frameName)** - This event is generated by WebXtra to request authorization to navigate to a given URL. You can use it to generally monitor all page navigation operations before they occur. It allows your scripts to restrict navigation to only certain sites, or intercept special URLs.

Your code should return true to allow the navigation to take place, or false to cancel it silently. Here are some examples:

*Lingo examples:*

```
on startNavigation spriteRef, URL, frameName
    -- Only load pages from the macromedia.com domain
    if not URL contains "macromedia.com" then
        alert("blocked")
        return false
    end if
    -- Alternatively, you could blocks a specific domain
    -- if URL contains "macromedia.com" then
    -- return false
    -- end if
    -- You can create your own messages in the HTML text as
```

well

```
    -- For example the page could contain a link that points
    to "lingo:reloadMovie"
```

```
    if URL contains "lingo:" then
```

```
        --use the do command to execute the "reloadMovie"
```

handler

```
        do URL.char[7..the number of chars in URL]
```

```
    return false
```

```

    end if
    --by default we allow all operations to complete
    return true
}
JavaScript syntax example:
function startNavigation(spriteRef, URL, frameName ){
    // Only load pages from the macromedia.com domain
    if (URL.indexOf("macromedia.com") < 0) {
        _player.alert("blocked")
        return false
    }
    // Alternatively, you could blocks a specific domain
    // if (URL.indexOf("macromedia.com") >= 0) {
    //     // return false
    // }
    // You can create your own messages in the HTML text
as well
    // For example the page could contain a link that points
to "lingo:reloadMovie"
    if (URL.indexOf("lingo:") == 0) {
        // parse the rest of the URL here, and pass it to the
do command for execution
        return false
    }
    //by default we allow all operations to complete
    return true
}

```

**openNewWindow(spriteRef)** - This handler is called by the Xtra sprite when the browser receives a request to create a new popup window. There are three possible return values:

- a) True (or 1) indicates that the operation should be allowed. The browser will create a standard, independent floating browser window, not controlled by the Xtra.
- b) False (or 0) blocks the creation of the new popup window.
- c) Advanced users can also instruct the browser to use an existing WebXtra browser window to handle the request. This is done by passing the "browserReference" property of any WebXtra sprite as a return value. The "Complete Browser" sample movie distributed with the Xtra has a full implementation of this behavior, where a new movie in a window is created to handle all popup requests. You can also return the the same browser window as the target of the popup operation, but some web sites do not handle this situation well and may generate Javascript errors in their pages. See the examples below:

*Lingo example:*

```

on openNewWindow spriteRef
    --if we want to reuse the existing browser sprite for the
new window:
    --return spriteRef.browserReference
    --
    --to allow the independent window to be created:
    --return true
    --
    --in this case we want to block all popups
    return false
end
JavaScript syntax example:

```

```

function openNewWindow (spriteRef) {
    //if we want to reuse the existing browser sprite for the
new window:
    //return spriteRef.browserReference
    //
    //to allow the independent window to be created:
    //return true
    //
    //in this case we want to block all popups
    return false
}

```

**closeWindowRequested(spriteRef)** - This handler is invoked when the browser receives a request to terminate the browser window, usually invoked by a Javascript call (`window.close()`) in the HTML code. WebXtra will never automatically destroy the window, but your code can schedule a window destruction if you want, specially for popup movies. It is important to notice that it is not recommended to quit the application or forget a window directly from this callback, as this interrupts normal message processing. Instead, it is recommended to set a flag or spawn a timeout that will destroy the window after the current handler has ended its execution. We recommend checking the [Complete Browser](#) sample available at the Tutorials page on our site for a proper implementation of window destruction.

*Lingo example:*

```

on closeWindowRequested spriteRef
    --sets a flag to indicate the window should be destroyed
    pMustDestroyWindow = TRUE
end

```

*JavaScript syntax example:*

```

function closeWindowRequested(spriteRef){
    //calls a routine to destroy the window
    scheduleWindowTermination()
}

```

**progressChange(spriteRef, progress, progressmax)** - This handler is called by WebXtra to signalize a network operation in progress. The two parameters `progress` and `progressmax` indicate how much of the operation has completed. A `progress` value of -1 or a `progressmax` value of 0 indicate that the operation has completed or was aborted.

*Lingo example:*

```

on progressChange spriteRef , progress, progressmax
    if progress>-1 and progressmax>0 then
        put string((float(progress)/float(progressmax)
* 100)) &"% completed"
    end if
end

```

*JavaScript syntax example:*

```

function progressChange(spriteRef, progress, progressmax){
    if ((progress>-1) && (progressmax>0)){
        trace((progress/progressmax*100.0).toString+"%
completed")
    }
}

```

**titleChange(spriteRef, newTitle)** - Called by WebXtra to signalize that the title of the document currently loaded has changed, usually when a page begins to load.

*Lingo example:*

```
on titleChane spriteRef , newTitle
    (the activeWindow).title = newTitle
end
```

*JavaScript syntax example:*

```
function titleChane(spriteRef, newTitle){
    _player.activeWindow.title = newTitle
}
```

**statusTextChange(spriteRef, newText)** - This handler is invoked when the text that needs to be displayed in the status bar has changed.

*Lingo example:*

```
on statusTextChange spriteRef , newText
    member("statustext").text = newText
end
```

*JavaScript syntax example:*

```
function statusTextChange(spriteRef, newText){
    member("statustext").text = newText
}
```

**navigateBackState(spriteRef, enabled)** - Called to indicate the current state of BACK navigation in the browser history, usually when a page finishes loading, or whenever necessary. Your code can use this information to enable or disable the BACK button in your interface.

*Lingo example:*

```
on navigateBackState spriteRef , enabled
    if enabled=true then
        sprite(8).membernum = member
("enabledbutton").membernum
    else
        sprite(8).member.membernum = member
("disabledbutton").membernum
    end if
end
```

*JavaScript syntax example:*

```
function navigateBackState(spriteRef, enabled){
    if (enabled==true) {
        sprite(8).membernum = member
("enabledbutton").membernum
    } else {
        sprite(8).member.membernum = member
("disabledbutton").membernum
    }
}
```

**navigateForwardState(spriteRef, enabled)** - Called to indicate the current state of FORWARD navigation in the browser history, usually when a page finishes loading, or whenever necessary. Your code can use this information to enable or disable the FORWARD button in your interface.

*Lingo example:*

```
on navigateForwardState spriteRef , enabled
    if enabled=true then
```

```

        sprite(9).membernum = member
("enabledbutton").membernum
    else
        sprite(9).member.membernum = member
("disabledbutton").membernum
    end if
end
JavaScript syntax example:
function navigateForwardState(spriteRef, enabled){
    if (enabled==true) {
        sprite(9).membernum = member
("enabledbutton").membernum
    } else {
        sprite(9).member.membernum = member
("disabledbutton").membernum
    }
}

```

**secureLockState(spriteRef, state)** - Called by WebXtra to inform the current state of the lock icon, indicating that a secure connection is in place. Valid state values are from 0 to 6. Values greater than 0 indicate that some sort of secure connection is in place. See the table below for the meaning of each value:

- 0 = Unsecured
- 1 = Mixed Environment (some elements are secure, others are not)
- 2 = Secure connection, unknown strength
- 3 = Secure connection using 40 Bit encryption
- 4 = Secure connection using 56 Bit encryption
- 5 = Secure connection using Fortezza encryption
- 6 = Secure connection using 128 Bit encryption

*Lingo example:*

```

on secureLockState spriteRef , state
    if state > 1 then
        put "Secure connection in place"
    end if
end

```

*JavaScript syntax example:*

```

function secureLockState(spriteRef, state){
    if (state > 1) {
        trace("Secure connection in place")
    }
}

```



## **WEBXTRA HELP: CREATING PROJECTORS**

WebXtra can be used to create projector in all operational systems and platforms supported by Director 8.5, Director MX and Director MX 2004.

**Important:** please remember that WebXtra sprites do not spawn a working web browser on the Macintosh: the Xtra is only fully functional on Windows. The Macintosh versions of the Xtra are provided as stubs to help users that must author cross-platform movies on the Mac platform, or need to work with their movies in both platforms with the same set of files and cast libraries, without any error messages.

### **CREATING A STANDARD WINDOWS OR MACOS PROJECTOR (Director 8.5 and MX)**

Director 8.5 and MX can only create native projectors. Windows projectors need to be created on a Windows machine, and Macintosh projectors need to be created on a Macintosh computer. However, Director movies (.dir files) containing WebXtra members don't need any modification in order to work on both platforms. A developer can work primarily on the Mac and only transfer the final .dir file to Director for Windows in order to create a Windows projector, or vice-versa. The only requirement is to install and register the Xtra on both platforms. Instructions for installing the Xtra can be found in the download packages. Please notice that WebXtra serial numbers are cross-platform: you can use the same serial number to register the software on both Mac and Windows.

After the Xtra is installed and registered, just select CREATE PROJECTOR from the FILE menu in Director, and choose a Director movie to be included in your projector.

The Xtra is automatically included in the Projector if your first movie contains a WebXtra cast member. If you are creating a "dummy" projector that will call your .dxr movie, you can include the Xtra in the Projector using the MODIFY->MOVIE->XTRAS menu, and adding the Xtra manually.

**TIP:** You can also deliver the WebXtra file in a folder named XTRAS, located in the same directory of your Projector, if you do not want to embed the Xtra file into your projector. This is recommended for faster startup of your program.

### **CREATING A CROSS-PLATFORM OR STANDARD PROJECTOR (Director MX 2004)**

Director MX2004 includes the ability to create Windows projectors when hosted on Mac OS X, and vice-versa. The OSX version of Director MX 2004 can create projectors for older versions of MacOS (8 and 9) as well. However, this only works correctly if Director is configured to locate and include the proper Xtra files for "the other" platform. Please make sure the appropriate files are installed in the Configuration\Cross platform resources Director folder according to the **installation** notes, and also make sure the Configuration\xtrainfo.txt file contains information about WebXtra (this procedure is also covered in the installation instructions) Assuming the cross-platform files are already installed and configured correctly you can invoke the FILE->PUBLISH SETTINGS menu item to

configure the parameters for your projector. The first step is to configure the FORMATS tab. The Mac version of Director MX 2004 can create separate projectors for Mac OSX, Classic and Windows, while the Windows version can create projectors for Windows and OSX only. You should disable the option to create a Shockwave file, as WebXtra is not available for Shockwave. In the FILES tab you can add additional Director movies to your projector. The Xtra will be automatically included if there is at least one WebXtra castmember in the main director movie. To finalize just click the PUBLISH button. Director MX 2004 saves the publishing settings with your Director movie, and future projectors can be created simply by choosing the FILE->PUBLISH menu item.



## **WEBXTRA HELP: HOW TO ORDER & REGISTER**

The unregistered version of WebXtra is fully-functional and may be used for evaluation, nonprofit and educational purposes only: commercial distribution is strictly prohibited. A registered version of WebXtra can be used in commercial products, and may be purchased online at [xtras.tabuleiro.com](http://xtras.tabuleiro.com), using a secure server. At our web site you can also access pages describing our purchase policies, purchase instructions, payment, delivery and security methods.

If you decide to buy the Xtra you don't need to download a new copy of the software. After your order is processed you will receive an e-mail with a serial number to register the software you've already installed on your machine.

To register you should double-click a WebXtra castmember and enter your serial number and registration information in the About Box that will be displayed. Click the REGISTER button to finish. Please keep your serial number archived. You will need it to register again if you reinstall Director or WebXtra.



## **WEBXTRA HELP: LICENSING & AVAILABILITY**

Web is a commercial product. Current price and updated information can be found at [xtras.tabuleiro.com](http://xtras.tabuleiro.com). If your product provides printed documentation and package we ask you to kindly include the following copyright information:

**WebXtra(tm) (c) Tabuleiro Prod Ltda 2004  
All Rights Reserved  
Xtra is a trademark of Macromedia, Inc.**

No royalty-fees are required for distribution of the Xtra with your projectors.



## **WEBXTRA HELP: TECHNICAL SUPPORT**

Please use the **Your Account** section available at our web site [xtras.tabuleiro.com](http://xtras.tabuleiro.com) to contact technical support. The site also contains Technotes and other resources that can help you identify and solve the most common problems quickly.