Mobile Internet Toolkit

Manuel-Jesús Prieto Martín [1]

[1] Telefónica Investigación y Desarrollo, Spain
mjprieto@telefonica.es

Resumen: En este capítulo se presenta el "mobile internet tooolkit", se introduce Flash Lite 1.1 y se analiza su potencial. También se se presenten ejemplos prácticos. Macromedia Flash Lite es una versión del repro-ductor de Macromedia Flash, diseñada y desa-rrollada con el objetivo de funcionar sobre dispositivos mó-viles, especialmente teléfonos mó-viles. Con este objetivo y teniendo en cuenta las importantes limitaciones que presentan este tipo de dispositivos, en cuanto a memoria, espacio de almacenamiento o capacidad de proce-so, Flash Lite presenta un compromiso de equilibrio entre estas limitaciones, y las caracterísi-cas y funcionalidades habituales en Macromedia Flash. Teniendo en cuenta la reciente aparición del producto y lo relativamente costoso que es conse-guir un estándar que los fabricantes instalen e integren en sus termi-nales, parece claro que, en el futuro, Macromedia Flash será una de las plataformas habituales de ejecución de aplicacio-nes dentro de los terminales móviles.

Palabras clave: Programación móvil; Macromedia Flash

Abstract. This chapter introduces the "mobile internet tooolkit", introduces Flash Lite 1.1 and analyses its potential. Practical examples are also presented. Macromedia Flash Lite is a version of the Macromedia Flash player, designed and developed to work on mobile devices, especially mobile phones. With this objective and taking into account the important limitations of this type of device, in terms of memory, storage space or process capacity, Flash Lite presents a compromise of balance between these limitations, and the usual features and functionalities of Macromedia Flash. Considering the recent appearance of the product and how relatively expensive it is to guide a standard that manufacturers install and integrate into their terminals, it seems clear that in the future, Macromedia Flash will be one of the usual platforms for running applications within mobile terminals.

Keywords: Mobile programming; Macromedia Flash

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

INTRODUCCIÓN A FLASH LITE 1.1

Macromedia Flash Lite es una versión del reproductor de Macromedia Flash, diseñada y desarrollada con el objetivo de funcionar sobre dispositivos móviles, especialmente teléfonos móviles. Con este objetivo y teniendo en cuenta las importantes limitaciones que presentan este tipo de dispositivos, en cuanto a memoria, espacio de almacenamiento o capacidad de proceso, Flash Lite presenta un compromiso de equilibrio entre estas limitaciones, y las características y funcionalidades habituales en Macromedia Flash. Actualmente existen dos versiones de Flash Lite, la versión 1.0 y la versión 1.1.

A grandes rasgos, y como idea genérica, podemos estructurar a Macromedia Flash Lite de la siguiente manera. Por una parte, proporciona un entorno de procesado de imágenes y recursos gráficos en general, que se encarga de gestionar y mostrar estos elementos en la pantalla del dispositivo. Como es habitual en Macromedia Flash, se soportan campos de texto para mostrar textos estáticos, dinámicos o como punto de entrada de texto por parte del usuario.

Ambas versiones de Flash Lite (1.0 y 1.1) soportan los formatos de audio típicos de los dispositivos móviles, como pueden ser MIDI o MFi, así como la gestión estándar de sonidos de Macromedia Flash.

También dispone de un interprete de *actionscript* que soporta la versión de este lenguaje usada en la versión 4 del reproductor estándar de Macromedia Flash, y que soporta además una serie de comandos específicos de la versión Lite, que permiten interactuar con algunas de las capacidades del dispositivo. Esta versión de actionscript, es denominada Flash Lite 1.x ActionScript.

Una característica importante que presenta Macromedia Flash Lite y que sin duda es adecuada para dispositivos móviles, es la conectividad de red. Con esta versión de Flash podremos cargar tanto ficheros *swf* externos como datos externos. También podremos realizar conexiones *http*, lo que sin duda aumenta de forma significativa la riqueza de las aplicaciones que se desarrollen sobre esta plataforma [1-5].

Junto con la conectividad, que es una característica básica de los dispositivos móviles y que Macromedia Flash Lite explota, también podremos aprovechar otras características como es el envío de mensajes cortos (SMS) o la realización de llamadas telefónicas.

Macromedia Flash Lite es soportado en la actualidad por un conjunto significativo de terminales móviles en el mercado. Teniendo en cuenta la reciente aparición del producto y lo relativamente costoso que es conseguir un estándar que los fabricantes instalen e integren en sus terminales, parece claro que en el futuro, Macromedia Flash será una de las plataformas habituales de ejecución de aplicaciones dentro de los terminales móviles. Hay varios puntos clave que apoyan esta idea y que el mercado no puede obviar:

- Cada día los terminales móviles tienen mayores capacidades y permiten ejecutar aplicaciones más complejas.

- Los usuarios demandan aplicaciones complejas, especialmente de entretenimiento, que doten de valor añadido sus terminales.

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

- Flash es una herramienta perfecta para desarrollar aplicaciones con un coste bajo y con unos resultados gráficos espectaculares. Este punto es importante en cualquier entorno, pero en el caso de los teléfonos móviles y el uso masivo de los mismos, cobra especial importancia.

- Es enorme la base de desarrolladores y diseñadores que conocen Macromedia Flash, y que por lo tanto pueden comenzar en un tiempo muy corto a crear aplicaciones que funcionen sobre dispositivos móviles.

- Los terminales móviles demandan aplicaciones sencillas, pero con unos resultados gráficos buenos. Teniendo en cuenta las limitaciones en la interfaz de usuario de los terminales (teclado y pantalla, principalmente), es obvio que un teléfono móvil no es el entorno adecuado para un procesador de texto o para un juego de estrategia 3D. Sin duda, es un entorno más adecuado para una aplicación de consulta de la meteorología gráficamente atractiva o para un juego sencillo y adictivo.

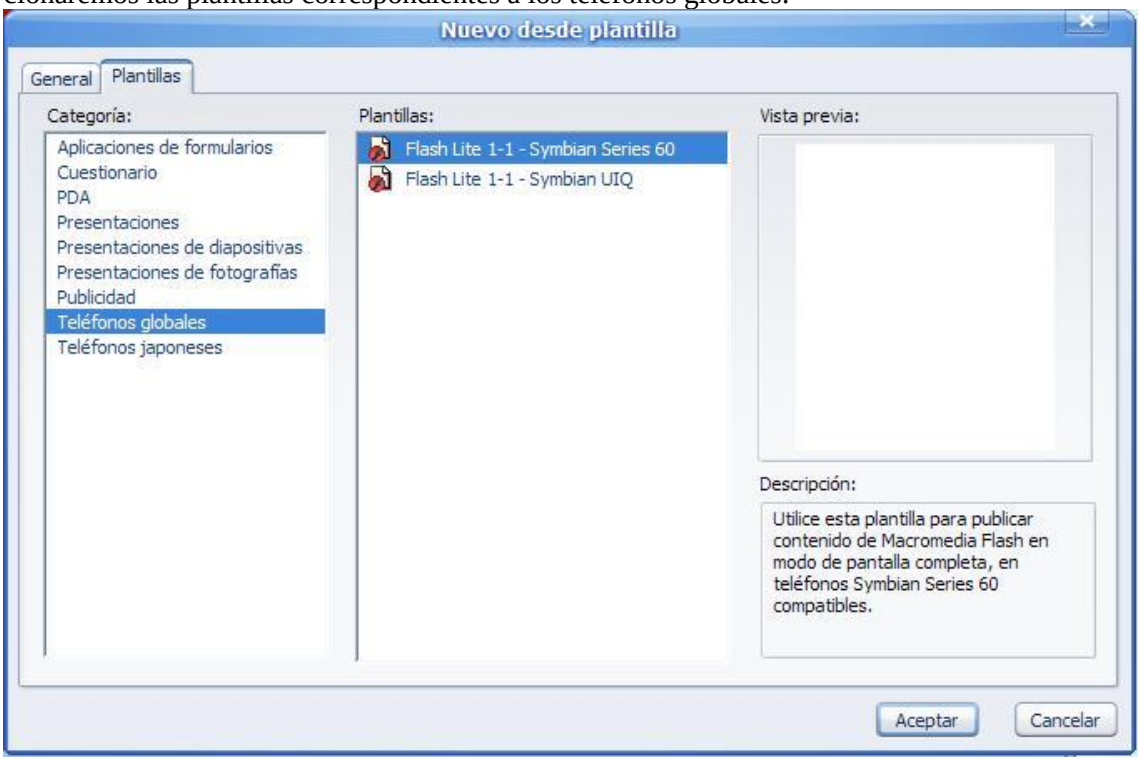
Actualmente el reproductor de Flash Lite se ejecuta dentro de los terminales como una aplicación independiente (*standalone*). De forma contrario a lo que ocurre en entorno de ordenadores habitual de Macromedia Flash, en el entorno de los terminales móviles, la heterogeneidad entre estos es un hecho y las características en las que difieren unos terminales de otros son muchas e importantes. Así, como ocurre en todos los desarrollos destinados a ser ejecutados en un teléfono móvil, el diseñador o desarrollador tiene que tener en cuenta qué familia de terminales será su objetivo de tal forma que el resultado final sea lo suficientemente bueno y que aproveche todas las capacidades y características del terminal. También se debe conocer y tener presente desde el primer momento el tipo de contenidos que el terminal o terminales objetivos permite o soporta. Por ejemplo, algunos dispositivos permiten realizar salvapantallas, otros permiten la integración de las aplicaciones Flash Lite dentro de páginas *web*, pero no todas estas características están presentes en todos los terminales. Este punto es especialmente importante porque en función del tipo de contenido y del propio dispositivo, dispondremos dentro de la aplicación de algunas capacidades. Así, una aplicación que es ejecutada como salvapantallas, no tendrá permiso para realizar conexiones de red. Para ayudar al desarrollador en este entorno tan diverso, la versión 8 de Macromedia Flash permite probar las aplicaciones sobre los diferentes entornos y así facilitar de forma importante este trabajo [6-11].

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241

LA PRIMERA APLICACIÓN: HOLA MUNDO

Como método de acercamiento al entorno de desarrollo que provee Macromedia Flash Professional 8 para las aplicaciones Flash Lite, vamos a realizar una primera aplicación. Esta aplicación tan sencilla, se limitará a mostrar por pantalla un texto y una imagen, ya que el principal objetivo es conocer el entorno de Macromedia Flash.

Para crear la aplicación comenzaremos por seleccionar la opción *Nuevo* en el menú de *Fichero*. En la pantalla que aparece seleccionaremos la plantilla a aplicar al nuevo documento Flash a crear, y seleccionaremos las plantillas correspondientes a los teléfonos globales.

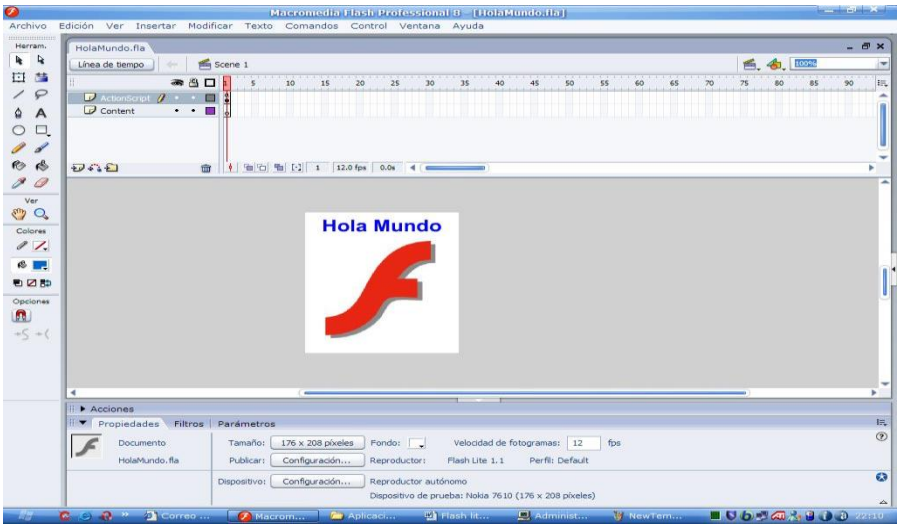


En esta pantalla nueva, y tal como se puede ver en la imagen anterior, seleccionamos la plantilla correspondiente a la serie 60 de Symbian. Así, se creará el nuevo documento y este estará preconfigurado y preparado para la plataforma que le hemos indicado. En este momento ya tenemos ante nosotros el entorno habitual de Macromedia Flash y podremos comenzar a trabajar de igual forma que cuando se desarrollan las aplicaciones Flash habituales.

En el espacio configurado para el documento, insertaremos el texto “Hola Mundo” y una imagen, que es el objetivo que nos hemos propuesto para esta primera aplicación. El resultado de estas acciones es el que muestra la siguiente imagen.

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241



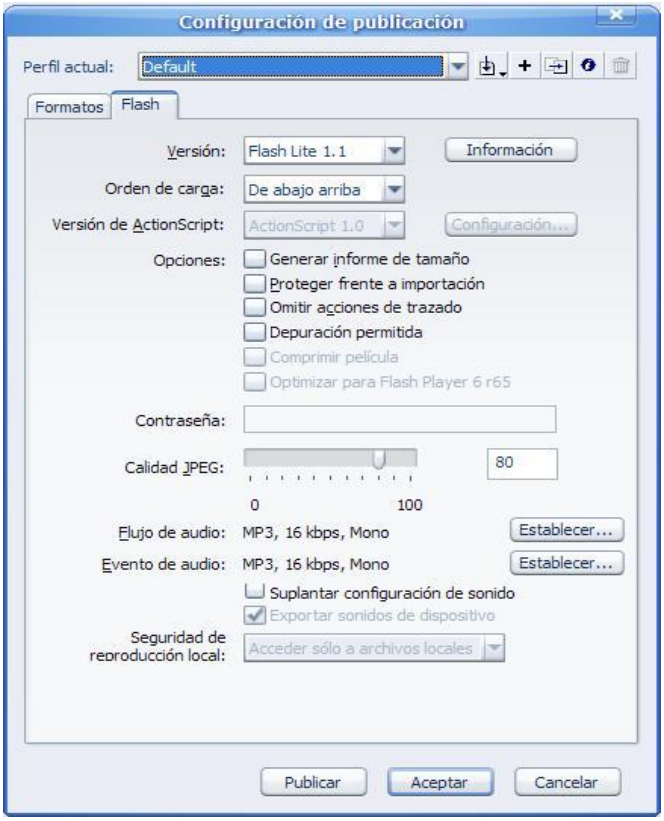
En este momento ya estamos dispuestos a probar la ejecución de nuestra aplicación dentro del entorno. Es obvio que la aplicación no hará nada y simplemente mostrará el contenido descrito por pantalla, pero es suficiente en estos momentos para nuestros propósitos.

Para probar la aplicación únicamente tendremos que seleccionar el menú Control y la opción Probar Película. Se abrirá una ventana que nos permite configurar el entorno en el que queremos probar la aplicación. La siguiente imagen muestra cómo sería la siguiente imagen.

En la parte izquierda de la pantalla, podemos seleccionar el dispositivo que queremos utilizar para probar la aplicación, dentro de la lista que se nos presenta.

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

Con estos pasos ya conocemos de qué forma podemos crear nuestras aplicaciones para Flash Lite y ya sabemos cómo probarlas. Como vemos, la herramienta Macromedia Flash Professional 8, nos ayuda de forma importante en esta labor [12-15].
A continuación, vamos a ver cómo podemos configurar de manera manual una aplicación, para Flash Lite y para un terminal concreto. Comenzaremos igual que antes, creando un documento nuevo, pero sin recurrir en este caso a las plantillas. Una vez creado el documento, iremos a la configuración de la publicación en el menú de fichero. En la ventana que aparece, dentro de la pestaña correspondiente a Flash, seleccionaremos la versión de flash que queremos utilizar como base para el nuevo documento que acabamos de crear.

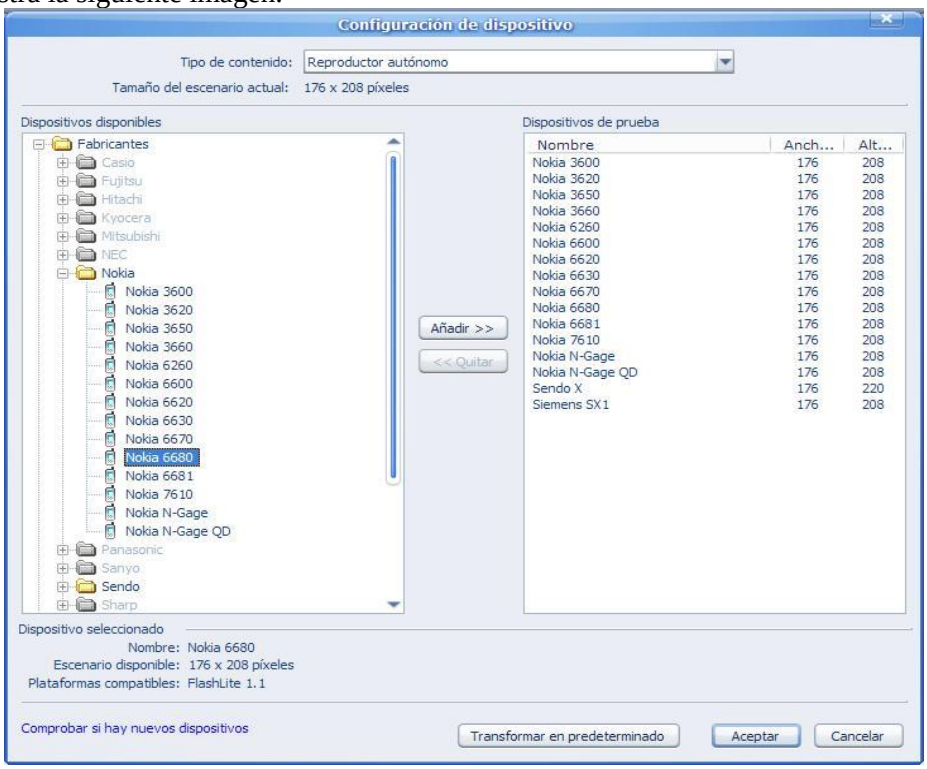


Una vez que volvemos a la pantalla principal, seleccionaremos la opción de configuración del dispositivo dentro del menú de propiedades de la película. Esta operación también se puede hacer a través del menú de fichero (*Fichero/Configuración de publicación*).

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

En la nueva pantalla que se abre para permitirnos la configuración del dispositivo, tendremos la posibilidad de seleccionar el tipo de contenido que queremos crear, que en nuestro caso será una aplicación independiente (*Reproductor anónimo*). En la lista de posibles dispositivos que se presenta, buscaremos aquel que deseemos y lo seleccionaremos. En la parte inferior de la pantalla, se nos informa de las características del dispositivo que hayamos seleccionado en cada momento. Esta información comprende el nombre del dispositivo, el tamaño de la pantalla y las versiones de la plataforma soportadas por el dispositivo. En nuestro caso seleccionaremos un terminal de la marca Nokia, tal y como muestra la siguiente imagen.



Es importante que el tamaño de nuestra película coincida con el tamaño de la pantalla del terminal, por lo que tendremos que cambiar el tamaño de la película a través del botón correspondiente en la ventana de propiedades.

Ahora podremos comenzar a trabajar en nuestro documento o película y podremos probarlo tal y como hicimos anteriormente.

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

GESTIÓN DE COMANDOS EN EL EMULADOR

El emulador que utilizaremos para el desarrollo de nuestras aplicaciones presenta algunas teclas de comando que podremos programar, a parte de las teclas alfanuméricas que tiene cualquier teléfono. Estas teclas son las que tienen todos los terminales, para descolgar o moverse por la pantalla, entre otros. La siguiente imagen muestra estas teclas. Las teclas alfanuméricas son obvias y las teclas especiales que usaremos en la mayoría de las aplicaciones, son las teclas de la fila superior, tanto la tecla grande con las cuatro pequeñas flechas (izquierda, derecha, arriba y abajo), como las teclas que están a cada lado de esta. A estas últimas las llamaremos teclas programables. Es importante destacar que el punto central de la tecla grande, será la tecla de selección y se utilizará de forma masiva en las aplicaciones para seleccionar botones.

En la mayoría de las aplicaciones, tendremos que programar alguna acción sobre las teclas, de tal forma que la aplicación responda a ciertas acciones por parte del usuario. Para gestionar la pulsación de una tecla por parte del usuario tendremos que controlar, tal y como es común en flash, el siguiente evento:

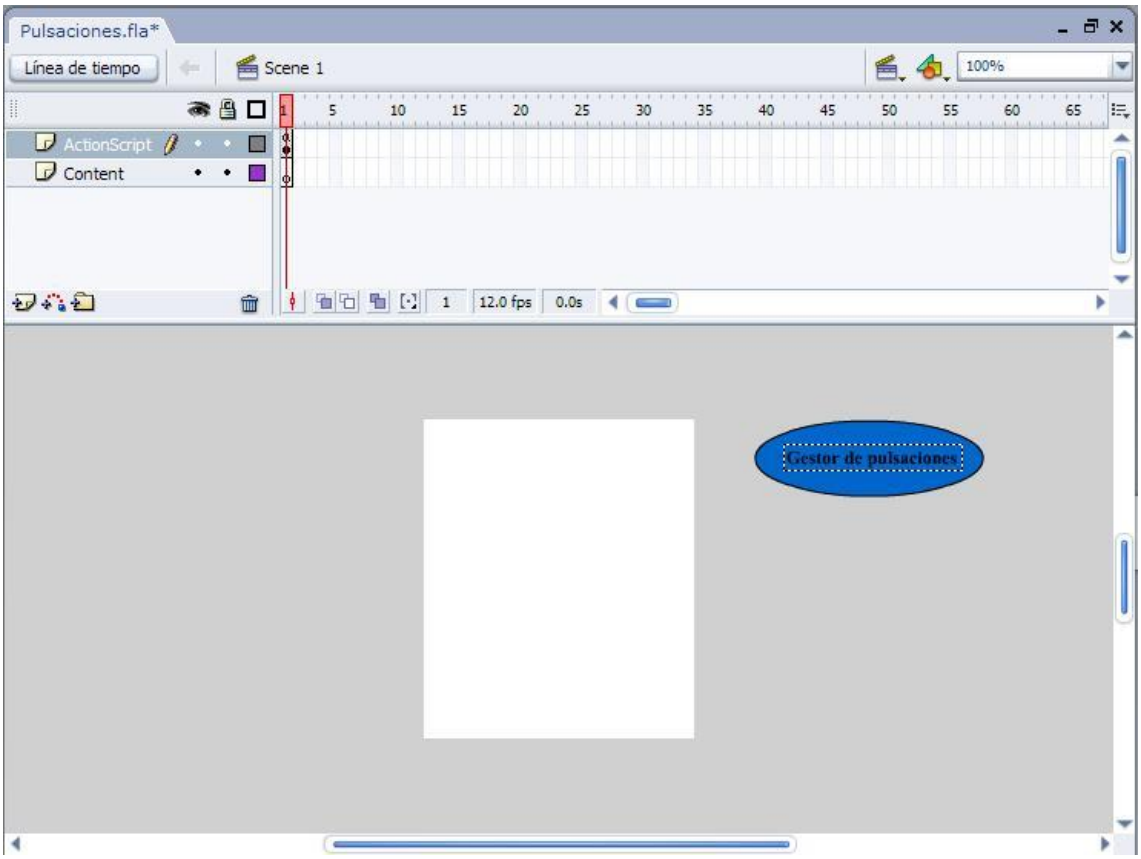
```
on(keyPress “<Right>")
```

Este caso es para recoger una pulsación de la tecla con la flecha hacia la derecha. Para el resto de teclas, la función a escribir es similar, pero cambiando el texto *<Right>* por otro, para indicar otra tecla. El siguiente ejemplo muestra como capturar los eventos de pulsación de teclas.

Para comenzar, y debido a que esta aplicación tiene como única finalidad el aprendizaje del modo en que se capturan las pulsaciones de las teclas, tendremos un solo botón en la aplicación, y nada más que este botón. Además, este botón no se mostrará por pantalla y nunca será visible. La siguiente imagen muestra esta situación [16-20].

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241



Como vemos, la pantalla está en blanco y el botón está fuera de dicha pantalla. Ahora lo único que haremos será añadir código a este botón, de tal manera que podamos comprobar de forma práctica cómo se capturan los eventos de teclado. El código que incluiremos en el botón se muestra en el listado a continuación.

```
on(keyPress "<Down>") {

        trace("Se ha pulsado la tecla de desplazamiento hacia abajo");

}

on(keyPress "<Up>") {

        trace("Se ha pulsado la tecla de desplazamiento hacia arriba");

}
```


ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

```
on(keyPress "<Left>"){
        trace("Se ha pulsado la tecla de desplazamiento hacia la izquierda");
}
on(keyPress "<Right>"){
        trace("Se ha pulsado la tecla de desplazamiento hacia la derecha");
}
on(keyPress "<Enter>"){
        trace("Se ha pulsado la tecla de selección");
}
on(keyPress "0"){
        trace("Se ha pulsado el 0");
}
on(keyPress "5"){
        trace("Se ha pulsado el 5");
}
on(keyPress "9"){
        trace("Se ha pulsado el 9");
}
on(keyPress "*"){
```

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

```
        trace("Se ha pulsado la tecla de asterisco");

}
```

Este código únicamente mostrará una traza en la ventana de salida del entorno de desarrollo de Flash, pero es suficiente para los objetivos didácticos de este sencillo ejemplo.

SetSoftKeys

El método *SetSoftKeys* nos permite modificar la asignación de las teclas programables del dispositivo. Estas teclas son las que se han comentado anteriormente y que se encontraban a ambos lados del teclado central. No todos los dispositivos nos permiten trabajar con estas teclas. Además, este método únicamente podrá utilizarse cuando el reproductor de Flash Lite se ejecute en modo autónomo. Es decir, cuando el reproductor se inicie dentro del contexto de otra aplicación, como puede ser un navegador, este comando no podrá ser invocado y por lo tanto estas teclas no serán aplicables.
Este comando recibe como parámetros dos textos, que serán los textos asociados a los botones programables y que podremos utilizar para capturar los eventos. El primer texto hacer referencia a la tecla programable izquierda y el segundo hace referencia a la tecla programable derecha. El método retornará un valor indicando si el comando es admitido o no es admitido en el terminal en el momento de ejecución. Este valor será 0 para indicar que está permitido y -1 para lo contrario [21-24].
El siguiente ejemplo muestra cómo activar las teclas programables y como utilizarlas. Para comenzar, crearemos un clip que represente un círculo azul y colocaremos dos en la pantalla del dispositivo. A uno de estos clips lo llamaremos *puntoizquierda* y al otro *puntoderecha*. Para comenzar, en la capa denominada *ActionScript*, incluiremos este código:

```
fscommand2("SetSoftKeys", "Izquierda", "Derecha");

fscommand2("FullScreen", "true");

_root.puntoizquierda._alpha = 0;

_root.puntoderecha._alpha = 0;
```

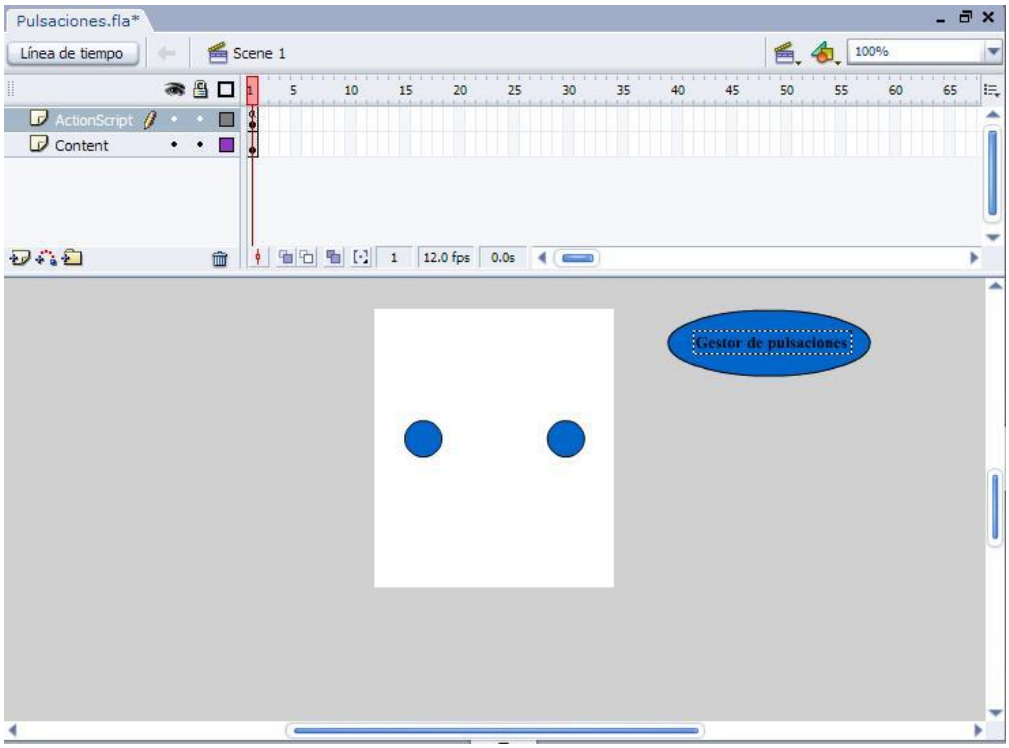
Este sencillo código activa las teclas programables, establece el modo pantalla-completa y hace invisibles los puntos de los que hemos hablado. También incluiremos un botón con el objetivo de controlar los eventos de ratón, igual que ya hicimos anteriormente. El código que asociaremos a este botón es:

```
on (keyPress "<PageUp>") {

        _root.puntoizquierda._alpha = 100;
```


ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

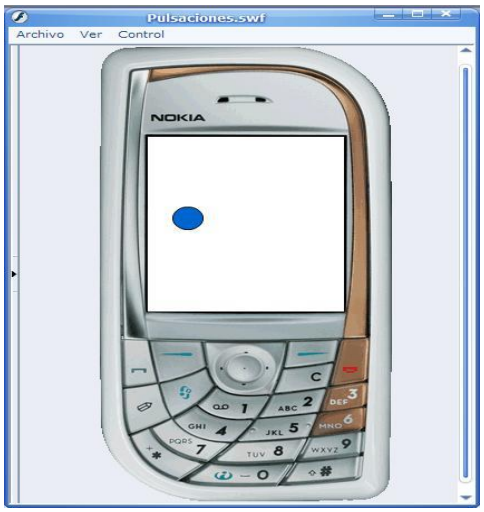
```
        _root.puntoderecha._alpha = 0;

}

on (keyPress "<PageDown>") {

        _root.puntoizquierda._alpha = 0;

        _root.puntoderecha._alpha = 100;

}
```

La tecla *PageUp* corresponde a la tecla programable de la izquierda y *PageDown* corresponde a la tecla programable de la derecha. El código lo que hace es mostrar y ocultar un punto u otro, según se pulse una u otra tecla programable. Las siguientes imágenes ilustran el proceso de desarrollo y ejecución.



ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241



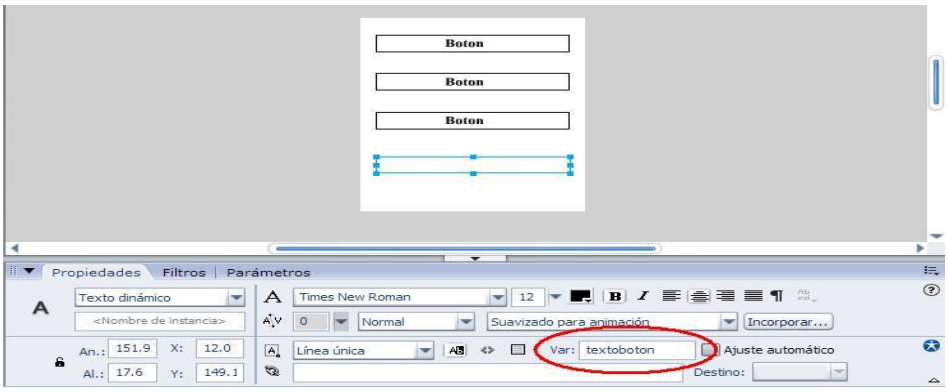
Uso de botones

Los botones dentro de las aplicaciones Flash Lite, funcionan de manera similar a como lo hacen en las aplicaciones flash clásicas. Tenemos los siguientes eventos a controlar en los botones:

- *press* – Se produce al pulsar la tecla de selección sobre el botón.
- *release* – Se produce cuando el usuario suelta la tecla de selección.
- *rollOver* – Se produce cuando el botón es seleccionado.
- *rollOut* – Se produce cuando el botón pierde la selección.

La siguiente aplicación muestra cómo utilizar botones dentro de las aplicaciones Flash Lite. Es una aplicación muy sencilla que presenta tres botones en la pantalla y un campo de texto dinámico, que mostrará el valor de una variable. El contenido de esta variable será modificado en función del botón que se pulse en cada momento. La variable se llama *textoboton* tal y como muestra la imagen siguiente.

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc



El código de los botones se muestra a continuación. El listado muestra el código de los tres botones, pero es evidente que cada una de las funciones está asociada a un botón.

```
// Botón superior

on(press){

        textoboton = "Botón superior";

}



//Botón intermedio

on(press){

        textoboton = "Botón intermedio";

}



// Botón inferior

on(press){

        textoboton = "Botón inferior";
```

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

```
}
```

Por último, vamos a incluir dos capturas de pantalla del emulador, para ver cómo sería la ejecución de esta sencilla aplicación.

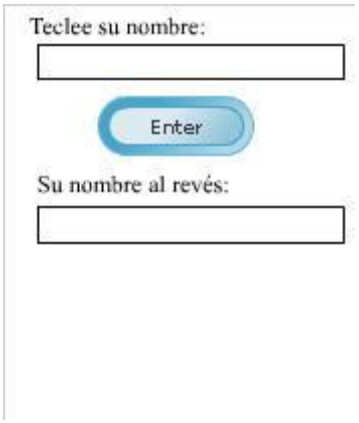
La figura anterior indica que se ha pulsado el botón intermedio, y por eso se muestra el texto al final de la pantalla, y también indica que en este momento, una pulsación de la tecla de selección actuaría sobre el botón inferior.

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

TEXTO EN FLASH LITE

Flash lite permite la utilización de campos de texto de tres modos diferentes. Por una parte, podremos utilizar textos estáticos, en los que, como es habitual en flash, se muestra un texto y este no puede ser modificado a lo largo de la ejecución de la película. Este tipo de campos de texto, son utilizamos comúnmente como sencillas etiquetas de texto. Los campos de texto de contenido dinámico, el segundo tipo de campo de texto, nos permite modificar el texto en tiempo de ejecución, a través de la asociación de este campo de texto a una variable [25-30]. El contenido de esta variable será el texto que se muestre en la pantalla. Este tipo de campos de texto lo hemos utilizado ya en el documento en alguno de los ejemplos. Por último, tenemos los campos de texto que permiten interactuar con el usuario, de tal forma que este pueda escribir en el campo de texto e introducir así información en la aplicación.

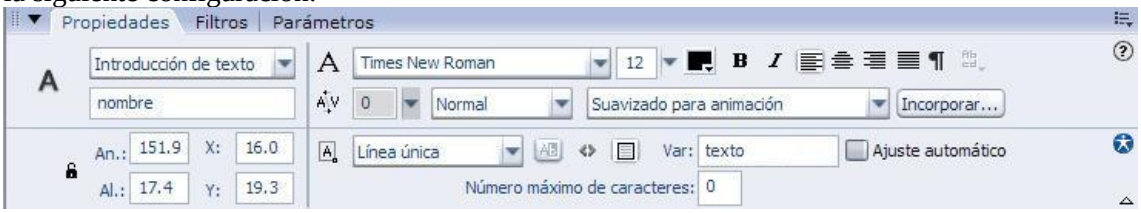
Para mostrar de forma práctica y sencilla cómo es la utilización de estos campos de texto y cuál es el cometido de cada uno de ellos, vamos a escribir un sencillo ejemplo. El ejemplo contiene los tres tipos de campos de texto:



Por una parte tenemos dos etiquetas de texto estático que corresponden con los textos:

- Teclee su nombre:
- Su nombre al revés:

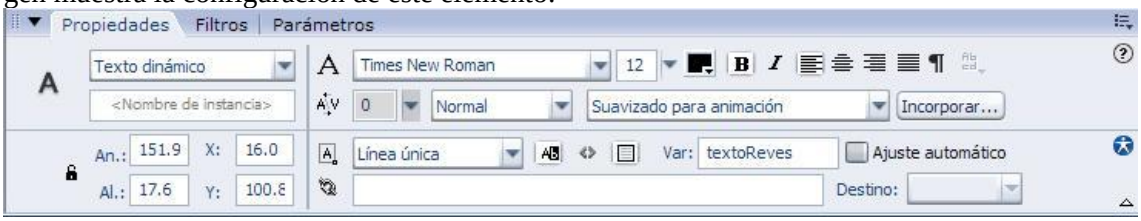
El campo de texto que está debajo de la primera etiqueta, es para introducir texto, tal y como muestra la siguiente configuración:



ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

Como vemos en la imagen, el campo de texto es del tipo “Introducción de texto”, y la variable a la que está asociado es texto. Es decir, el texto que escriba el usuario en este elemento, será almacenado dentro de la variable texto. Esta variable no necesita ser definida en ningún sitio.
El campo de texto de la parte inferior de la pantalla, es un campo de texto dinámico, y su cometido es mostrar el contenido de una variable. Antes hemos visto como introducir información en una variable a través de un campo de texto, y ahora vemos el proceso contrario. Este campo de texto, mostrará el contenido de la variable pero no permitirá que se modifique dicha variable. La siguiente imagen muestra la configuración de este elemento:



Como vemos, el tipo es “Texto dinámico” y la variable a la que está asociado este campo de texto es “textoReves”.
Para completar el ejemplo, tenemos un botón con el texto “Enter”, que tendrá el código actionscript que nos permite aportar un poco de lógica al ejemplo. En concreto, el objetivo de esta sencilla aplicación es tomar un texto introducido por el usuario, darle la vuelta a dicho texto, y mostrarlo en el campo de texto correspondiente a través de la variable textoReves. El código actionscript del botón es el siguiente:

```
on(press){

	resultado = "";

	for (i=length(texto);i>0; i--){

		resultado = resultado add substring(texto, i, 1);

	}

	textoReves = resultado;

}
```

Como vemos, el código es muy sencillo. Simplemente define una variable llamada “resultado”, en la que va añadiendo las letras de la cadena original (texto), empezando por la última y finalizando en la primera. Por último, se introduce el resultado final en la variable textoReves.
La siguiente imagen muestra el resultado final en el emulador.

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

Restricciones en los campos de introducción de texto

Se pueden configurar los caracteres que el usuario puede introducir dentro de un campo de texto. A través del comando *SetInputTextType* se puede realizar este trabajo. Así, por ejemplo, si quisiéramos que en el ejemplo anterior, el usuario únicamente pudiera escribir números dentro del campo de texto, tendríamos que escribir el siguiente comando dentro de la aplicación:

```
fscommand2("SetInputTextType", "texto", "Numeric");
```

Si escribimos esta línea e intentamos introducir algún carácter no numérico dentro del campo de texto, tendremos un aviso por parte del emulador, tal y como muestra la siguiente imagen.

Los posibles modos que se pueden indicar en el control de los caracteres son los que se listan a continuación:

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

- Numeric – Únicamente se permiten números (0-9).
- Alpha – Sólo se permiten caracteres alfabéticos (a-z, A-Z).
- Alphanumeric – Caracteres alfanuméricos (0-9, a-z, A-Z).
- Latin – Caracteres latinos, es decir, los alfanuméricos y los símbolos de puntuación.
- NonLatin – Únicamente permite los caracteres no latinos.
- NoRestriction – Es el modo predeterminado, sin limitación alguna.

El método fscommand2 mostrado anteriormente, devuelve una variable que nos permite comprobar si la orden se ha ejecutado correctamente o no. Esto es debido a que no todos los dispositivos admiten la gestión de estas restricciones y por lo tanto, a través de esta variable podremos comprobar si nuestra orden tendrá efecto o si por el contrario será ignorada por el terminal. Los posibles valores de la variable son:

- 0 – En caso de error
- 1 – Si la orden se ha ejecutado correctamente y por lo tanto la restricción tendrá efecto.

Fuentes de texto

En Flash Lite disponemos de varias formas para representar las fuentes de los campos de texto. La primera y más sencilla es aplicar alguna de las fuentes de texto que están disponibles de forma genérica:

- _sans
- _serif
- _typewriter

Aún así, el resultado final depende en cierta medida del dispositivo final, pero en cualquier caso este tratará de adecuarse a lo indicado en la aplicación. Esta configuración se realiza a través del menú de propiedades del campo de texto. Tendremos que seleccionar el tipo de fuentes a utilizar y la fuente concreta.

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

El menú de selección del tipo de texto tiene cinco opciones, pero únicamente tres están disponibles para Flash Lite.

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

SONIDOS

Dentro de la versión Lite de Flash tenemos dos tipos de sonido:

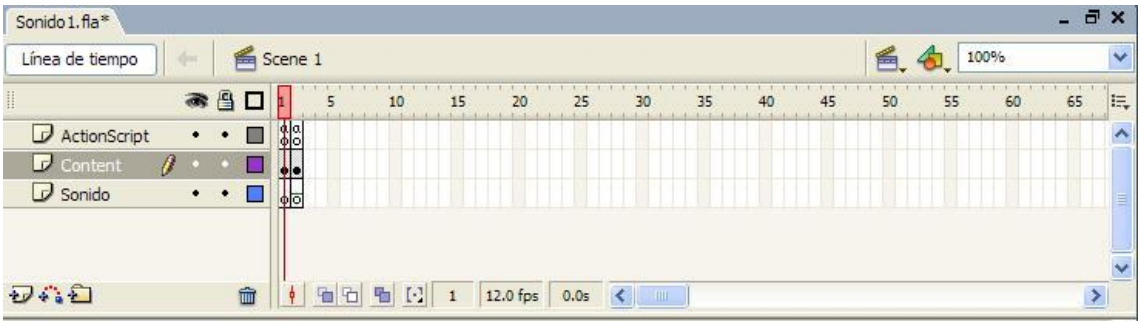
- Sonidos de dispositivo.
- Sonidos estándar o nativo.

Los sonidos de dispositivo se guardan dentro del fichero publicado (con extensión swf) y su formato es el nativo del dispositivo (MIDI o MFi). El sonido será reproducido por el dispositivo, ya que Flash se lo pasa a este para su descodificación y reproducción. En cuanto a las limitaciones de reproducción de este tipo de sonidos, debemos tener en cuenta que Flash Lite no puede sincronizar las animaciones y el sonido del dispositivo y que la versión 1.0 únicamente admite este tipo de sonidos, sonidos de dispositivo. La reproducción de este tipo de sonidos es responsabilidad del dispositivo o terminal, y no del propio entorno Flash, lo que limita sus capacidades [31-25].
En cuanto a los sonidos estándar o nativos únicamente están disponibles en la versión 1.1, tal y como se indicó anteriormente. Flash Lite permite sincronizar este tipo de sonidos con las animaciones, lo que provee a este tipo de sonidos de mucha más utilidad que los sonidos de dispositivo. Además de los formatos de ficheros de sonido que ya hemos comentado antes y que son admitidos como tipos de sonido de dispositivo, los formatos admitidos en Flash Lite 1.1 son SMAF, PCM, WAV, ADPCM y MP3.
Otra clasificación que podemos realizar en los sonidos, son sonidos de evento y sonidos de flujo. Los primeros se reproducen independientemente de la línea de tiempo y se reproducen hasta el final del sonido o hasta que la reproducción se detiene mediante ActionScript. Estos sonidos deben descargarse de forma completa antes de comenzar a reproducirse y una vez más tenemos que decir que su utilidad en las aplicaciones es menor que la de los sonidos de flujo [36-38].
El segundo tipo de sonidos, los sonidos de flujo, están sincronizados con la línea de tiempo y se utilizan para reproducir sonidos sincronizados con la animación. El sonido se detiene cuando la línea de tiempo se detiene y Flash Lite omite fotogramas de la animación si es necesario, para mantener la sincronización. En este caso, sincronizando la línea de tiempo y los sonidos, la utilidad en las aplicaciones es mucho mayor, ya que se puede controlar cuándo comienza a reproducirse el sonido y cuando se para, siempre en sincronía con la línea de tiempo y por tanto con la evolución de la aplicación.
A continuación vamos a realizar un sencillo ejemplo, que nos permitirá comprender de forma más práctica y rápida el trabajo con los sonidos dentro de Flash Lite.

Ejemplos de uso de sonidos

Lo primero que haremos para realizar este ejemplo con sonidos, será modificar la línea de tiempo, para insertar una nueva capa que llamaremos Sonido y extender la línea de tiempo de las capas a dos fotogramas. La siguiente imagen muestra esta estructura:

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc



Una vez que tenemos esta estructura de capas, insertaremos el siguiente código en el primer fotograma de la capa denominada ActionScript:

```
fscommand2("FullScreen", true);

stop();
```

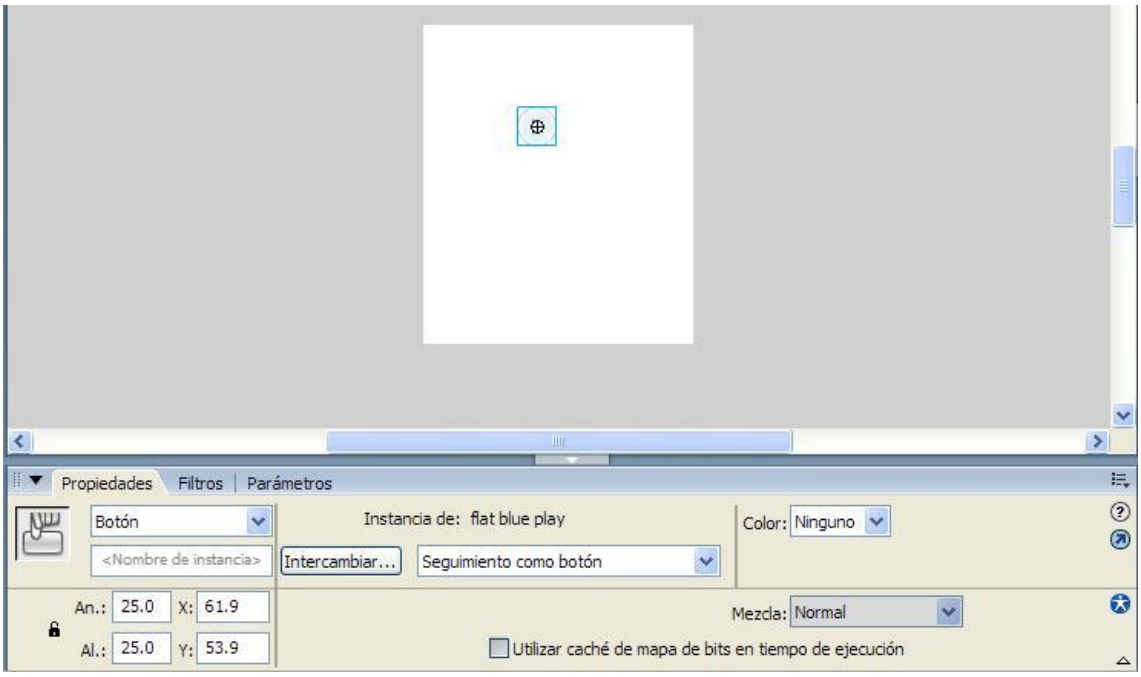
En el segundo fotograma de esta misma capa insertaremos la siguiente línea:

```
stop();
```

Con estas líneas de código simplemente paramos la línea de tiempo de ejecución de la aplicación, en cada fotograma. De esta forma, podemos pasar del fotograma uno al dos y viceversa y así tendremos dos pantallas dentro de la aplicación.
Las dos pantallas de la aplicación, se configuran en la siguiente capa, la que se denomina Content. La siguiente imagen muestra el contenido de este primer fotograma de la capa Content:

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241



La capa contiene únicamente un botón, tal y como muestra la imagen, que pertenece a la librería estándar de Flash, y que corresponde con el botón de play estándar de cualquier equipo de sonido. Este botón nos permite avanzar hasta la siguiente pantalla, el fotograma dos, con el siguiente código insertado en dicho botón:

```
on(press){

        gotoAndPlay(2);

};
```

Al pasar al segundo fotograma, comenzará a sonar música. Este proceso de gestión del sonido, lo veremos un poco más adelante, pero antes vamos a ver el contenido del segundo fotograma de la capa denominada Content. La siguiente imagen muestra dicho contenido:

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

El botón que contiene esta capa, es también un botón de la librería de botones estándar de Flash y corresponde con el botón de stop de cualquier equipo de sonido. Junto con el botón, tenemos un campo de texto estático que muestra el mensaje: "reproduciendo sonido".
El código que debe ser insertado en el botón, será el siguiente:

```
on(press){

        stopAllSounds();

        gotoAndPlay(1);

}
```

Como vemos, este botón lo que hace en primer lugar es parar la reproducción de cualquier sonido que esté en ese momento siendo reproducido, y a continuación vuelve a la primera pantalla de la aplicación, que ya hemos explicado anteriormente.
Para finalizar, vamos a ver qué serie de operaciones debemos realizar para conseguir que el sonido comience a reproducirse cuando pulsemos en el botón correspondiente en el primer fotograma. Lo primero que debemos hacer es insertar el sonido a reproducir en la aplicación. En nuestro caso, deseamos reproducir un sonido MIDI, y será un sonido de dispositivo.
Para disponer de este fichero MIDI dentro de nuestra aplicación, primero tendremos que insertar dentro de la biblioteca lo que denominaremos un sonido proxy, que será un sonido nativo, ya que Flash Lite no permite importar sonidos de dispositivo directamente. El formato de los sonidos proxy

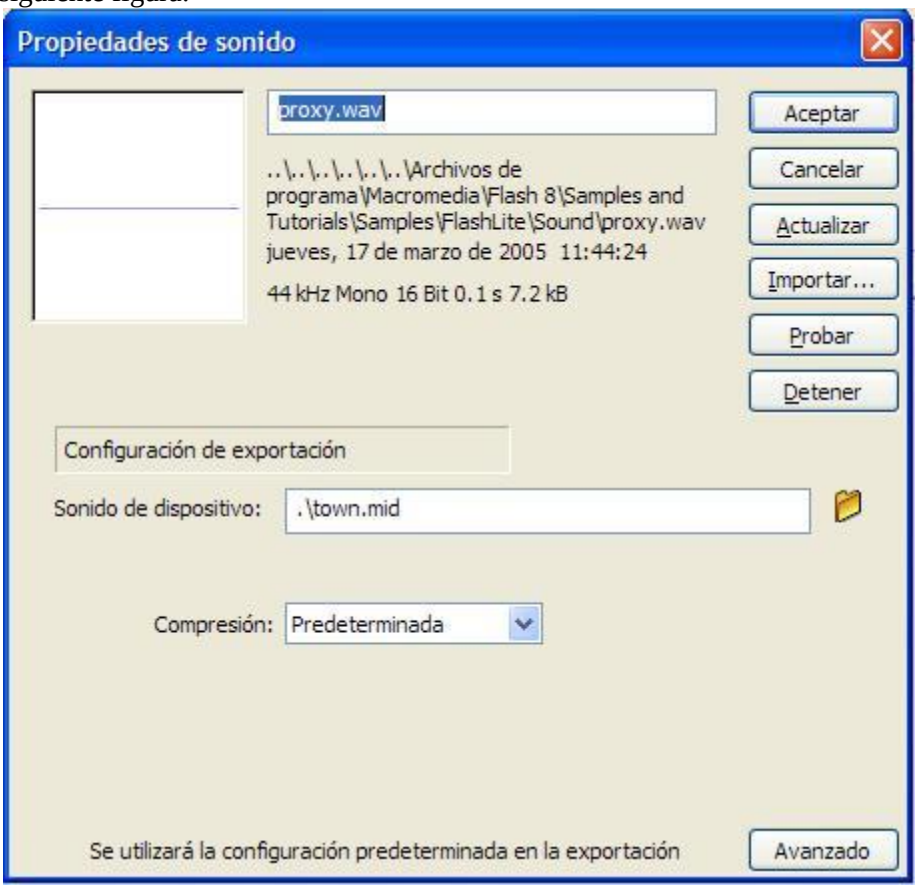
ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

será mp3, wav o aiff. En nuestro caso, el sonido proxy que vamos a utilizar es el fichero proxy.wav, que se distribuye junto con la aplicación Flash (versión 8), como parte de los ejemplos correspondientes a Flash Lite. En concreto, podemos encontrar este fichero en el directorio:

[Directorio_instalación]\Samples and Tutorials\Samples\FlashLite\Sound

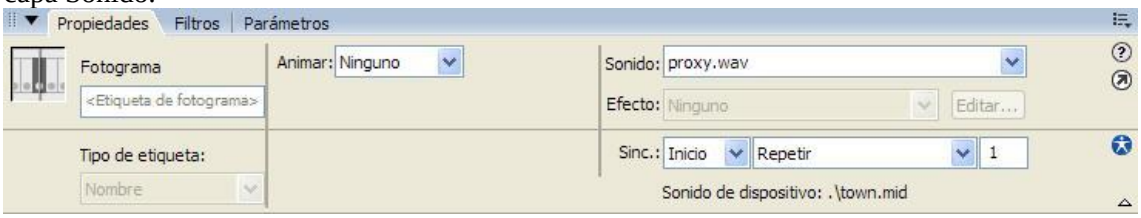
Insertamos este fichero dentro de la biblioteca de la aplicación, a través de la opción Archivo/Importar/Importar a biblioteca.
Una vez que tenemos el sonido de proxy insertado en la biblioteca, procederemos a asociar este sonido al sonido de dispositivo real que utilizaremos en nuestra aplicación. Tal y como hemos dicho anteriormente, este fichero a reproducir, será un fichero MIDI. Para realizar la asociación, abriremos la ventana de propiedades del elemento proxy.wav dentro de la ventana de la biblioteca de la aplicación y asociaremos el fichero MIDI a través del campo denominado Sonido de dispositivo, tal y como muestra la siguiente figura:



En nuestro caso, hemos utilizado el fichero town.mid. El lector, podrá seleccionar cualquier otro fichero MIDI que tenga accesible en su máquina. A partir de este momento, ya tenemos podremos utilizar el fichero proxy, para manejar el sonido dentro de la aplicación.

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

Para realizar la configuración final de la reproducción del sonido dentro de la aplicación, volveremos a la línea de tiempo. En el segundo fotograma de la capa denominada Sonido, asociaremos el sonido a esta capa, de tal forma que comience a reproducirse al entrar en este fotograma [39].
Debido a que estamos trabajando con un sonido de dispositivo, que tal y como hemos indicado, se reproducen de forma ininterrumpida hasta el final una vez que son activamos, tendremos que utilizar la orden stopAllSounds de ActionScript para detener la reproducción del sonido. Esta orden, está insertada en el botón de parar tal y como se mostró.
La siguiente imagen muestra cómo está asociada la reproducción del sonido con el fotograma 2 de la capa Sonido.



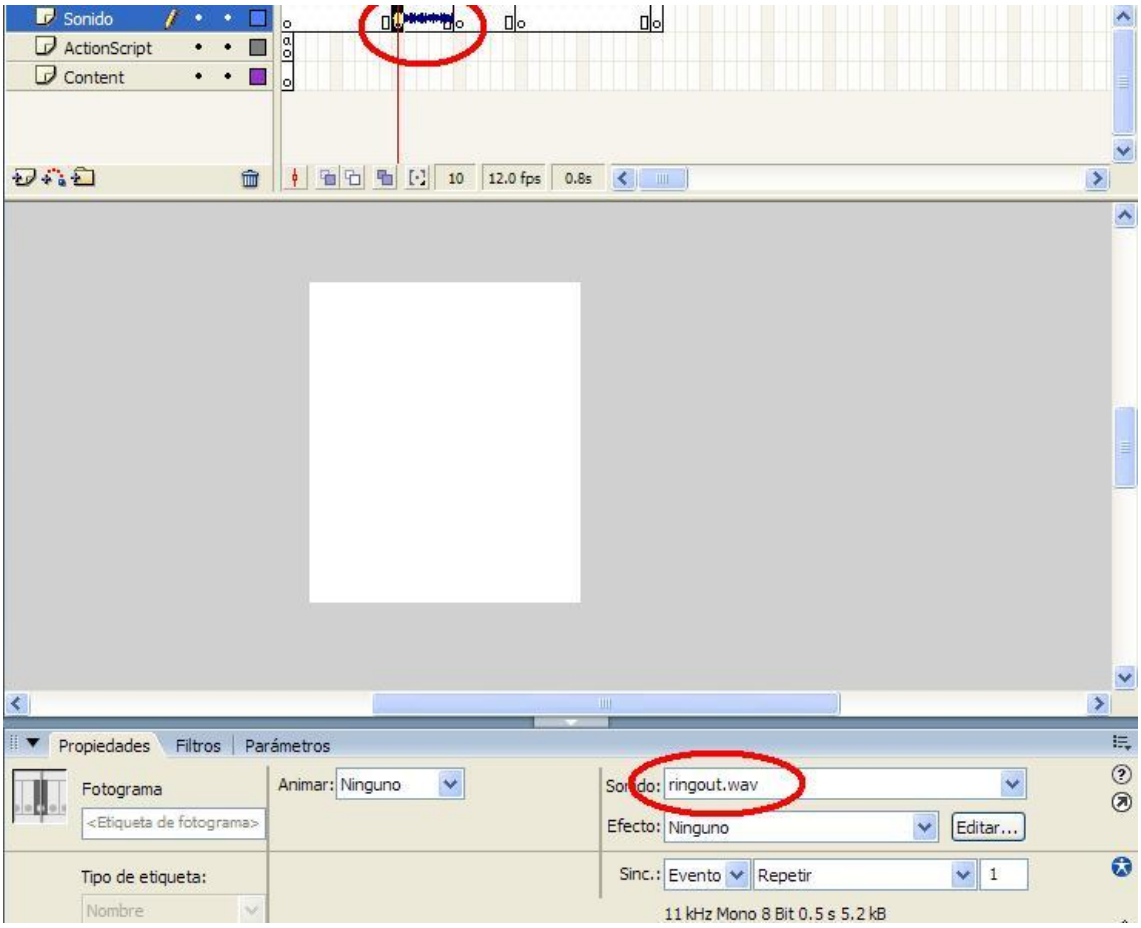
En la parte inferior derecha de la ventana de propiedades de la capa, tal y como se ve en la figura, tenemos el sonido (proxy.wav) y la acción.
Por último, y para completar el ejemplo, simplemente tendremos que ejecutar la aplicación seleccionando como dispositivo de ejecución un dispositivo que soporte el sonido. Las siguientes imágenes muestran la aplicación en ejecución.

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

Para utilizar sonidos nativos dentro de una aplicación, lo primero que deberemos hacer será importar el fichero en cuestión a la librería de la aplicación. En nuestro caso, hemos usado un fichero wav que insertaremos en una capa de la línea de tiempo, denominada Sonido, a través de la ventana de propiedades de la capa, tal y como hemos hecho en el ejemplo anterior. Una vez hecho esto, sencillamente con la gestión de los fotogramas de esta capa, tendremos la posibilidad de sincronizar el sonido con la an imación.

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc



ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

ACTIONSCRIPT EN FLASH LITE 1.1

Aunque ya hemos hecho algún uso de actionscript dentro de Flash Lite, a continuación vamos a ver de forma más detallada cuáles son las capacidades del lenguaje dentro de esta versión de Flash y sus posibles usos.
La versión de actionscript disponible en la versión 1.1 de Flash Lite contiene ciertos elementos presentes en la versión del lenguaje actionscript que incorporaba Flash 4, junto con ciertos elementos específicos de Flash Lite, propios del tipo de dispositivos al que está orientada la plataforma: iniciar llamadas, mensajes de texto…
Hay algunos elementos correspondientes al lenguaje actionscript de la versión 4 de Flash, que no están presentes en la versión Lite:

- Las funciones startDrag y stopDrag.
- La propiedad _dropTarget.
- La propiedad _soundBufTime.
- La propiedad _url.
- La propiedad de conversión String().

Junto con estas restricciones, la gestión de pulsaciones de botones y teclas también difiere en cierta medida, pero ya hemos visto como trabajar con estos elementos [40].
Algunos otros puntos que conviene resaltar como ausentes en Flash Lite, debido a su presencia en las versiones para PC son:

- Funciones definidas por el usuario.
- Matrices, objetos y otros tipos de datos complejos.
- La carga en tiempo de ejecución de archivos externos de imagen y sonido.

Para especificar la ruta a un determinado clip o línea de tiempo, se utiliza la sintaxis de barras (/), tal y como se utiliza en la gestión de directorios en sistemas de ficheros. También se pueden utiliza dos puntos (..) y los elementos _levelN, _root y _parent. Con estos elementos, conseguiremos seleccionar aquel elemento de la línea de tiempo al que queremos dirigirnos. Por ejemplo:

```
tellTarget("/prueba/clip1") {
```


ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

```
   _x = 100;

}
```

Este código cambiaría el valor de la propiedad _x del clip denominado clip1, que está contenido en el clip prueba, que a su vez está contenido en la línea de tiempo principal de la película.
Para utilizar variables dentro del actionscript, tendremos que utilizar una sintaxis similar a la descrita en el párrafo anterior, usando también los dos puntos (:). También se puede utilizar una notación basada en puntos. Es decir, la variable var1 del elemento clip1 del que hablábamos en el párrafo anterior, Tendríamos que poner cualquiera de estas dos líneas:

```
/prueba/clip1/:var1

_root.prueba.clip1.var1
```

Debido a la restricción en el uso de matrices de la que hablábamos, tendremos que gestionar este tipo de estructuras de datos de otra forma. Una emulación del mismo es lo que tenemos a continuación. Para hacerlo utilizaremos la función eval, que permite acceder a las variables, propiedades o clips de película por su nombre. Así, aunque con ciertas limitaciones sobre el uso de matrices estándar tal y como se hace en la versión para Pc de Flash, podremos resolver en cierta medida el problema utilizando un determinado formato para el nombre de las mismas. Tendremos que definir los datos como:

```
dato_0 = “valor de la posición 1”;

dato_1 = “valor de la posición 2”;

dato_2 = “valor de la posición 3”;

dato_3 = “valor de la posición 4”;
```

Como podemos ver, los nombre de las variables que componen lo que podríamos denominar la matriz virtual, tienen un parte del nombre común y al final tienen un número que podríamos identificar con el índice del dato dentro de la matriz. Para hacer referencia ahora a estos datos almacenados, tendríamos que usar un fragmento de código similar al siguiente, teniendo en cuenta que queremos acceder a la posición determinada por el valor de la variable pos, dentro de la matriz virtual:

```
eval (“dato_” add pos);
```

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

Para mostrar todo el contenido de la matriz en la ventana de salida de la aplicación, podríamos usar lo siguiente:

```
for (i = 0; i <4; i++) {

   trace (eval ("dato_" add i));

}
```

También podríamos crear variables nuevas, e ir así aumentando el tamaño de nuestra matriz virtual. Por ejemplo, si tenemos una variable tam_dato que contiene en tamaño de la matriz virtual en cada momento, podríamos añadir un nuevo elemento a la matriz con el siguiente fragmento de código:

```
eval (“dato_” add tam_dato) = “Valor nuevo”;

tam_dato++;
```

Con este sencillo truco, podemos utilizar matrices en nuestras aplicaciones de una forma sencilla y efectiva, a pesar de que Flash Lite no soporte este tipo de datos de forma nativa.
Otra restricción de la que hemos hablado antes y que nos encontramos en Flash Lite, es la utilización de funciones personalizadas. Esta restricción se puede también subsanar de una forma sencilla a través de la función call(). Esta función nos permite la ejecución del código insertado en un determinado fotograma, sin necesidad de desplazar la línea de tiempo hasta el mismo. A través de esta función se puede invocar un fotograma de la misma línea de tiempo, o un fotograma de cualquier clip de la aplicación. De esta forma, insertando código en ciertos fotogramas y utilizando la función call para invocar dicho código, podremos simular el uso de funciones personalizadas [41].

Elementos propios del actionscript de Flash Lite 1.1

A continuación vamos a ver aquellas características propias del lenguaje actionscript para la versión Lite de Flash. Comenzaremos por ver cómo se inicia una llamada telefónica, un mensaje de texto o un mensaje multimedia:

```
getURL(“tel:123456789”);

getURL(“sms:123456789”);

getURL(“sms:123456789?body=Texto del mensaje. Saludos”);

getURL(“mms:123456789”);
```


ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

Dos elementos importantes dentro del actionscript de Flash Lite y que ya hemos visto en algún momento a lo largo del documento son fscommand y fscommand2. El primero de ellos nos permite lanzar otras apliciones:

```
status = fscommand("launch",

  "d:\\system\\apps\\browser\\browser.app,http://www.usal.es");

}
```

Como vemos, el formato del segundo parámetro de la función, contiene en primer lugar el nombre de la aplicación a lanzar, seguido de los parámetros que desean pasarse a esta aplicación, separados por “,”.
La instrucción fscommand2 es similar a fscommand en su sintaxis. También devuelve un valor y recibe unos parámetros. Estos parámetros son un comando en primer lugar y separados por comas del comando y entre sí, la información adicional que este comando necesite. Los comandos que se le pueden pasar a fscommand2 son los siguientes:

- Escape – Codifica una cadena de texto para ser enviada a través de la red, sustituyendo los caracteres no alfanuméricos por su secuencia de escapa hexadecimal.

- FullScreen – Establece el tamaño de visualización de la aplicación, para que ocupe la pantalla completa o no (true o false). Sólo funciona cuando el reproductor de Flash Lite se ejecuta en modo autónomo, evidentemente, cuando el reproductor está embebido dentro de otra aplicación, no se puede aplicar.

- GetBatteryLevel – Esta llamada retorno el nivel de batería del que dispone el dispositivo en ese momento. El valor irá desde 0 hasta el valor máximo, que se puede averiguar a través de la llamada a fscommand2 con el comando GetMaxBatteryLevel.

- GetDateDay – Devuelve el día correspondiente a la fecha actual, es decir, puede tomar valores entre 1 y 31.

- GetDateMonth – Devuelve el mes correspondiente a la fecha actual, es decir, puede tomar valores entre 1 y 12.

- GetDateWeekday – Devuelve un valor numérico, correspondiente al día de la semana correspondiente a la fecha actual:

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

  - 0 – Domingo
  - 1 – Lunes
  - 2 – Martes
  - 3 – Miércoles
  - 4 – Jueves
  - 5 – Viernes
  - 6 – Sábado
  - -1 – Error

- GetDateYear – Devuelve el año correspondiente a la fecha actual.

- GetDevice – Indica a través de una variable cuyo nombre es una cadena de texto que se pasa como parámetro, el nombre identificador del dispositivo que está ejecutando la aplicación. Un ejemplo de uso de este comando se presenta a continuación y la variable devicename, sería la que contendría el valor una vez realizada la llamada:

```
fscommand2("GetDevice", "devicename");
```

- GetDeviceId – Indica a través de una variable cuyo nombre es la cadena de texto que se pasa como parámetro, un identificador único del dispositivo en el que se está ejecutando la aplicación, como puede ser el número de serie.

- GetFreePlayerMemory – Devuelve la cantidad de memoria disponible para Flash Lite, expresada en kilobytes.

- GetLanguage – Indica a través de una variable cuyo nombre es la cadena de texto que se pasa como parámetro, el idioma utilizado por el dispositivo.

- GetLocaleLongDate – Indica a través de una variable cuyo nombre es la cadena de texto que se pasa como parámetro, la fecha actual, expresada en formato largo a través de una cadena de texto.

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

- GetLocaleShortDate – Indica a través de una variable cuyo nombre es la cadena de texto que se pasa como parámetro, la fecha actual, expresada en formato corto a través de una cadena de texto.

- GetLocaleTime – Indica a través de una variable cuyo nombre es la cadena de texto que se pasa como parámetro, la hora actual.

- GetMaxBatteryLevel – Retorna el nivel máximo de batería.

- GetMaxSignalLevel – Retorna el nivel máximo de intensidad de señal.

- GetMaxVolumeLevel – Retorna el nivel máximo de volumen.

- GetNetworkConnectStatus – Indica el estado actual de la conexión de red. Los posibles valores que puede retornar esta llamada son:

    - 0 – Hay una conexión activa.

    - 1 – El dispositivo está conectándose.

    - 2 – No hay conexión activa.

    - 3 – Se ha interrumpido la conexión de red.

    - 4 – No de puede determinar el estado de la conexión de red.

    - -1 – Comando no admitido por el terminal.

- GetNetworkName – Indica a través de una variable cuyo nombre es la cadena de texto que se pasa como parámetro, el nombre de la red.

- GetNetworkRequestStatus – Retorna el estado de la última petición http realizada. Los posibles valores son:

    - 0 – Hay una solicitud pendiente, se ha establecido una conexión de red, se ha resuelto el nombre del servidor y se ha realizado conectado con este.

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

  - 1 – Hay una solicitud pendiente y se ha establecido una conexión de red.
  - 2 – Hay una solicitud pendiente, pero todavía no se ha establecido una conexión de red.
  - 3 – Hay una solicitud pendiente, se ha establecido una conexión de red y se está resolviendo el nombre de host del servidor.
  - 4 – La solicitud ha fallado debido a un error de la red.
  - 5 – La solicitud ha fallado debido a un error de conexión al servidor.
  - 6 – El servidor ha devuelto un error de HTTP (por ejemplo, 404).
  - 7 – La solicitud ha fallado debido a un error al acceder al servidor DNS o al resolver el nombre del servidor.
  - 8 – La solicitud se ha realizado correctamente.
  - 9 – La solicitud ha fallado debido a un error de tiempo límite agotado.
  - 10 – La solicitud aún no se ha realizado.
  - -1 – Comando no admitido.

- GetNetworkStatus – Devuelve un valor indicando el estado de la red telefónica. Los posibles valores son:
  - 0 – No hay ninguna red registrada.
  - 1 – Red doméstica.
  - 2 – Red doméstica ampliada.
  - 3 –Lejos de la red doméstica.
  - -1 – No se admite el comando.

- GetPlatform – Indica a través de una variable cuyo nombre es la cadena de texto que se pasa como parámetro, la descripción extendida del dispositivo.

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

- GetPowerSource – Indica el tipo de alimentación que se está usando. Los posibles valores son:
    - 0 – Batería.
    - 1 – Fuente externa de alimentación.
    - -1 – Comando no admitido.
- GetSignalLevel – Indica la intensidad de la señal a través de un valor numérico que va desde 0 hasta el valor determinado por el comando GetMaxSignalLevel.
- GetTimeHours – Indica las horas en el hora actual, y sus valores van desde 0 hasta 23.
- GetTimeMinutes – Indica los minutos en la hora actual, y sus valores van desde 0 hasta 59.
- GetTimeSeconds – Indica los segundos en la hora actual, y sus valores van desde 0 hasta 59.
- GetTimeZoneOffset – Indica a través de una variable cuyo nombre es la cadena de texto que se pasa como parámetro, el número de minutos de diferencia entre la hora local y la universal.
- GetTotalPlayerMemory – Indica la cantidad total de memoria, expresada en kilobytes.
- GetVolumeLevel – Indica el nivel actual de volumen. Este valor puede variar entre 0 y el valor indicado por GetMaxVolumeLevel.
- Quit – Este comando ordena la parada de la reproducción y cierra el reproductor. Sólo es aplicable cuando el reproductor funciona en modo autónomo.
- ResetSoftKeys – Este comando reestablece la configuración original de las flechas programables.
- SetInputTypeText – Este comando establece el modo en que debe abrirse el campo de introducción de texto, cuyo nombre recibe como segundo parámetro (el primero es el nombre del propio comando). El tercer parámetro indica el modo en cuestión y puede ser uno de los siguientes valores:
    - Numeric – 0-9.

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

    - Alpha – a-z y A-Z.
    - AlphaNumeric – 0-9, a-z y A-Z.
    - Latin – Alfanuméricos y puntuación.
    - NonLatin – Sólo caracteres no latinos.
    - NoRestriction

- SetQuality – Establece la calidad de la presentación. Sus valores pueden ser:
    - high
    - medium
    - low

- SetSoftKeys – Permite modificar la asignación de las teclas programables. Este comando se explicó con detalle en un fragmento anterior del documento.

- StartVibrate – Ordena al dispositivo que comience a vibrar. Se puede indicar el tiempo (en milisegundos) que el terminal debe estar vibrando y las veces que debe repetir la vibración, así como el tiempo que debe parar entre repeticiones.

- StopVibrate – Ordena al dispositivo que detenga la vibración, si está activa.

- Unescape – Decodifica una cadena de texto codificada anteriormente.

A continuación, vamos a ver las capacidades y variables de la plataforma:

- _capCompoundSound – Indica si Flash Lite puede procesar ficheros de sonido compuesto. En caso afirmativo su valor será 1 y undefined en caso contrario.

- _capEmail – Indica si se pueden enviar mensajes de correo electrónico a través del método getURL. En caso afirmativo su valor será 1 y undefined en caso contrario.

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

- _capLoadData – Indica si se pueden cargar datos externos a través de las funciones loadMovie, loadMovieNum, loadVariables y loadVariablesNum. En caso afirmativo su valor será 1 y undefined en caso contrario.

- _capMFi – Indica si el dispositivo puede reproducir sonidos en formato MFi. En caso afirmativo su valor será 1 y undefined en caso contrario.

- _capMIDI - Indica si el dispositivo puede reproducir sonidos en formato MIDI. En caso afirmativo su valor será 1 y undefined en caso contrario.

- _capMMS - Indica si se pueden enviar mensajes multimedia a través del método getURL. En caso afirmativo su valor será 1 y undefined en caso contrario.

- _capMP3 - Indica si el dispositivo puede reproducir sonidos en formato MP3. En caso afirmativo su valor será 1 y undefined en caso contrario.

- _capSMAF - Indica si el dispositivo puede reproducir archivos multimedia en formato SMAF. En caso afirmativo su valor será 1 y undefined en caso contrario.

- _capSMS - Indica si se pueden enviar mensajes de texto (sms) a través del método getURL. En caso afirmativo su valor será 1 y undefined en caso contrario.

- _capStreamSound – Indica si el dispositivo puede reproducir flujos de sonido. (sincronizado). En caso afirmativo su valor será 1 y undefined en caso contrario.

- _cap4WayKeyAS – Indica si actionscript ejecuta código asociado a las teclas de flecha (derecha, izquierda, arriba y abajo). En caso afirmativo su valor será 1 y undefined en caso contrario.

- $version – Es un texto que indica el número de versión de Flash Lite.

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241

References

1. Lucas Fernando Souza De Castro, Gleifer Vaz Alves, André Pinz Borges (2017). Using trust degree for agents in order to assign spots in a Smart Parking. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 6, n. 2
2. Rafhael Cunha, Cleo Billa, Diana Adamatti (2017). Development of a Graphical Tool to integrate the Prometheus AEOlus methodology and Jason Platform. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 6, n. 2
3. Jaime Rincón, Jose Luis Poza, Juan Luis Posadas, Vicente Julián, Carlos Carrascosa (2016). Adding real data to detect emotions by means of smart resource artifacts in MAS. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 5, n. 4
4. Christian Paulo Villavicencio, Silvia Schiaffino, J. Andrés Díaz-Pace, Ariel Monteserin (2016). A Group Recommendation System for Movies based on MAS. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 5, n. 3
5. Asset Management System through the design of a Jadex Agent System (2016). Javier Carbó, José M. Molina, Miguel A. Patricio. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 5, n. 2
6. Xiomara Patricia Blanco Valencia, M. A. Becerra, A. E. Castro Ospina, M. Ortega Adarme, D. Viveros Melo, D. H. Peluffo Ordóñez (2017). Kernel-based framework for spectral dimensionality reduction and clustering formulation: A theoretical study.
7. Juan Bullón, Angélica González Arrieta, Ascensión Hernández Encinas, Araceli Queiruga Dios (2017). Manufacturing processes in the textile industry. Expert Systems for fabrics production. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 6, n. 1
8. David Griol, Jose Manuel Molina (2016). Simulating heterogeneous user behaviors to interact with conversational interfaces. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 5, n. 4
9. Casado-Vara, R., & Corchado, J. (2019). Distributed e-health wide-world accounting ledger via blockchain. Journal of Intelligent & Fuzzy Systems, 36(3), 2381-2386.
10. Gustavo Isaza, Maria H. Mejía, Luis Fernando Castillo, Adriana Morales, Nestor Duque (2012). Network Management using Multi-Agents System. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 1, n. 3
11. Ana Cristina Bicharra, Nayat Sanchez-Pi, Luis Correia, José Manuel Molina (2012). Multi-agent simulations for emergency situations in an airport scenario. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 1, n. 3
12. Valérian Guivarch, Valérie Camps, André Péninou (2012). AMADEUS: an adaptive multi-agent system to learn a user's recurring actions in ambient systems. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 1, n. 3
13. Ichiro Satoh (2012). Bio-inspired Self-Adaptive Agents in Distributed Systems. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal (ISSN: 2255-2863), Salamanca, v. 1, n. 2
14. Román, J. A., Rodríguez, S., & de da Prieta, F. (2016). Improving the distribution of services in MAS. Communications in Computer and Information Science (Vol. 616). https://doi.org/10.1007/978-3-319-39387-2_4
15. Buciarelli, E., Silvestri, M., & González, S. R. (2016). Decision Economics, In Commemoration of the Birth Centennial of Herbert A. Simon 1916-2016 (Nobel Prize in Economics 1978): Distributed Computing and Artificial Intelligence, 13th International Conference. Advances in Intelligent Systems and Computing (Vol. 475). Springer.
16. González-Briones, A., De La Prieta, F., Mohamad, M., Omatu, S., & Corchado, J. (2018). Multi-agent systems applications in energy optimization problems: A state-of-the-art review. Energies, 11(8), 1928.
17. Prieto, J., Alonso, A. A., de la Rosa, R., & Carrera, A. (2014). Adaptive Framework for Uncertainty Analysis in Electromagnetic Field Measurements. Radiation Protection Dosimetry, ncu260.
18. Chamoso, P., Raveane, W., Parra, V., & González, A. (2014). Uavs Applied to the Counting and Monitoring Of Animals. In Advances in Intelligent Systems and Computing (Vol. 291, pp. 71–80). https://doi.org/10.1007/978-3-319-07596-9_8

ISBN: 978-84-9012-861-9
Colección Aquilafuente: AQ-241
Ediciones Universidad de Salamanca - cc by nc dc

19. Baruque, B., Corchado, E., Mata, A., & Corchado, J. M. (2010). A forecasting solution to the oil spill problem based on a hybrid intelligent system. Information Sciences, 180(10), 2029–2043. https://doi.org/10.1016/j.ins.2009.12.032
20. Tapia, D. I., & Corchado, J. M. (2009). An ambient intelligence based multi-agent system for alzheimer health care. International Journal of Ambient Computing and Intelligence, v 1, n 1(1), 15–26. https://doi.org/10.4018/jaci.2009010102
21. Mata, A., & Corchado, J. M. (2009). Forecasting the probability of finding oil slicks using a CBR system. Expert Systems with Applications, 36(4), 8239–8246. https://doi.org/10.1016/j.eswa.2008.10.003
22. Glez-Peña, D., Díaz, F., Hernández, J. M., Corchado, J. M., & Fdez-Riverola, F. (2009). geneCBR: A translational tool for multiple-microarray analysis and integrative information retrieval for aiding diagnosis in cancer research. BMC Bioinformatics, 10. https://doi.org/10.1186/1471-2105-10-187
23. Fernández-Riverola, F., Díaz, F., & Corchado, J. M. (2007). Reducing the memory size of a Fuzzy case-based reasoning system applying rough set techniques. IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, 37(1), 138–146. https://doi.org/10.1109/TSMCC.2006.876058
24. Méndez, J. R., Fdez-Riverola, F., Díaz, F., Iglesias, E. L., & Corchado, J. M. (2006). A comparative performance study of feature selection methods for the anti-spam filtering domain. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4065 LNAI, 106–120.
25. Corchado, J. M., Pavón, J., Corchado, E. S., & Castillo, L. F. (2004). Development of CBR-BDI agents: A tourist guide application. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 3155, pp. 547–559). https://doi.org/10.1007/978-3-540-28631-8
26. Laza, R., Pavn, R., & Corchado, J. M. (2004). A reasoning model for CBR_BDI agents using an adaptable fuzzy inference system. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 3040, pp. 96–106). Springer, Berlin, Heidelberg.
27. Corchado, J. A., Aiken, J., Corchado, E. S., Lefevre, N., & Smyth, T. (2004). Quantifying the Ocean's CO2 budget with a CoHeL-IBR system. In Advances in Case-Based Reasoning, Proceedings (Vol. 3155, pp. 533–546).
28. Corchado, J. M., Borrajo, M. L., Pellicer, M. A., & Yáñez, J. C. (2004). Neuro-symbolic System for Business Internal Control. In Industrial Conference on Data Mining (pp. 1–10). https://doi.org/10.1007/978-3-540-30185-1_1
29. Corchado, J. M., Corchado, E. S., Aiken, J., Fyfe, C., Fernandez, F., & Gonzalez, M. (2003). Maximum likelihood hebbian learning based retrieval method for CBR systems. In Lecture Notes in Computer Science (including subseries LNAI and Lecture Notes in Bioinformatics) (Vol. 2689, pp. 107–121). https://doi.org/10.1007/3-540-45006-8_11
30. Casado-Vara, R., Chamoso, P., De la Prieta, F., Prieto J., & Corchado J.M. (2019). Non-linear adaptive closed-loop control system for improved efficiency in IoT-blockchain management. Information Fusion.
31. González-Briones, A., Chamoso, P., Yoe, H., & Corchado, J. M. (2018). GreenVMAS: virtual organization-based platform for heating greenhouses using waste energy from power plants. Sensors, 18(3), 861.
32. Casado-Vara, R., Novais, P., Gil, A. B., Prieto, J., & Corchado, J. M. (2019). Distributed continuous-time fault estimation control for multiple devices in IoT networks. IEEE Access.
33. Chamoso, P., González-Briones, A., Rivas, A., De La Prieta, F., & Corchado J.M. (2019). Social computing in currency exchange. Knowledge and Information Systems.
34. Casado-Vara, R., Prieto-Castrillo, F., & Corchado, J. M. (2018). A game theory approach for cooperative control to improve data quality and false data detection in WSN. Int. J. of Robust and Nonlinear Control, 28(16), 5087-5102.
35. Chamoso, P., Rodríguez, S., de la Prieta, F., & Bajo, J. (2018). Classification of retinal vessels using a collaborative agent-based architecture. AI Communications, (Preprint), 1-18.
36. Chamoso, P., González-Briones, A., Rodríguez, S., & Corchado, J. M. (2018). Tendencies of technologies and platforms in smart cities: A state-of-the-art review. Wireless Communications and Mobile Computing, 2018.
37. Gonzalez-Briones, A., Prieto, J., De La Prieta, F., Herrera-Viedma, E., & Corchado, J. M. (2018). Energy Optimization Using a Case-Based Reasoning Strategy. Sensors (Basel), 18(3), 865-865. doi:10.3390/s18030865
38. Gonzalez-Briones, A., Chamoso, P., De La Prieta, F., Demazeau, Y., & Corchado, J. M. (2018). Agreement Technologies for Energy Optimization at Home. Sensors (Basel), 18(5), 1633-1633. doi:10.3390/s18051633
39. Chamoso, P., González-Briones, A., Rivas, A., De La Prieta, F., & Corchado, J. M. (2019). Social computing in currency exchange. Knowledge and Information Systems, 1-21.
40. Casado-Vara, R., Vale, Z., Prieto, J., & Corchado, J. (2018). Fault-tolerant temperature control algorithm for IoT networks in smart buildings. Energies, 11(12), 3430.

ISBN: 978-84-9012-861-9

Colección Aquilafuente: AQ-241