Capitulo 6

Como echarle el muerto a alguien.

Una de las cosas más importantes en una programación es tener un sistema que permite echarle el muerto a alguien. Dicho de manera elegante, el sistema debe identificar qué usuario se encuentra activado en cada momento, almacenando dichos valores en un histórico, indicando el instante tanto de su activación como de su desactivación, lo que posteriormente se puede consultar y presentar en un informe (o sea, lo que había dicho antes).

Quién no ha tenido que oír la famosa frase "pues no sé que pasó pero se estropeó", seguida de un "¿eso qué puede ser?". ¿A que siempre te han entrado ganas de contestar: "¡que no tienes ni idea ni de tocar el teclado, manazas!", en lugar del socorrido "tendré que mirarlo". Como siempre que ocurre algo de este estilo casualmente todo el mundo estaba en el "tigre", gracias al registro de la activación de los usuarios vamos a descubrir que más de uno ha alcanzado ya el don de la doble ubicuidad.

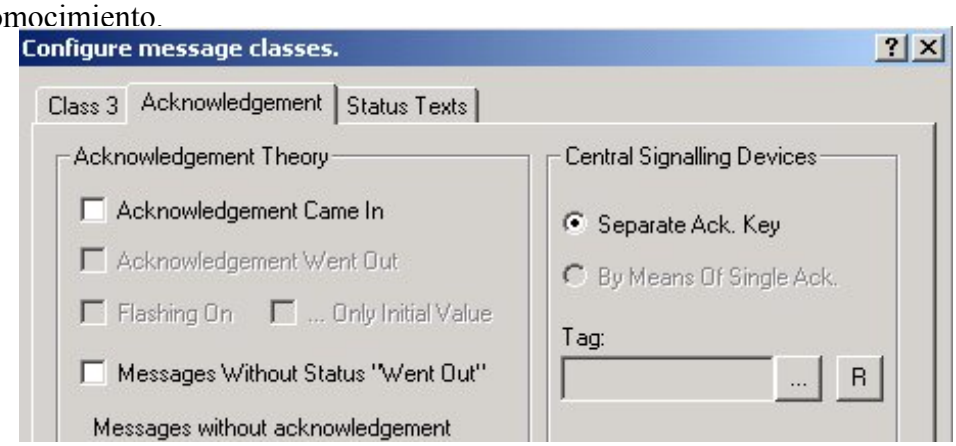
El WinCC no tiene un sistema de almacenamiento de usuarios. Bien, empezamos bien. Otra vez a inventárnosla. Pues puestos a inventar vamos a tratar a un usuario como si fuese una alarma, ya que eso sí que lo almacena el WinCC. Todos los usuarios van a ser la misma alarma ¿?, que se activará por una variable interna de tipo bool, que a su vez se activará por el login o logout del usuario. Fácil ¿no?.

Si has contestado sí, no sigas leyendo, listo. Para los que hayáis contestado no ahí va la explicación de la falla.

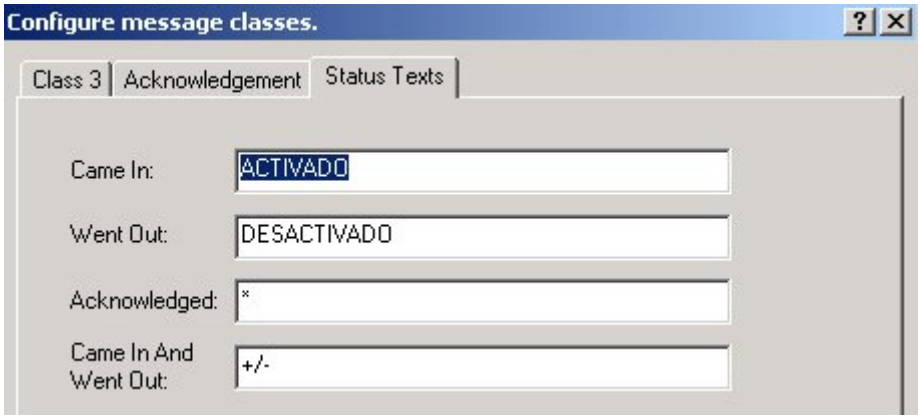
Primeramente nos hacemos una variable interna de tipo bool que vamos a llamar USUARIOS. A continuación vamos al alarm logging y creamos un nuevo process value block, dentro de blocks, que llamaremos USUARIOS, y que posee unos 20 caracteres.

En message classes vamos a crear una nueva clase, que llamaremos (¿lo adivinas?) usuarios, dentro del cual hacemos un nuevo message type llamado...¡usuarios! (que original). Asignamos un color identificativo para logarse (CAME in) y deslogarse (came out). Los otros da igual, ya que no vamos a acusar a los usuarios, ni desaparecerán sin ser acusados (esta última frase es el persé de todo el capítulo).

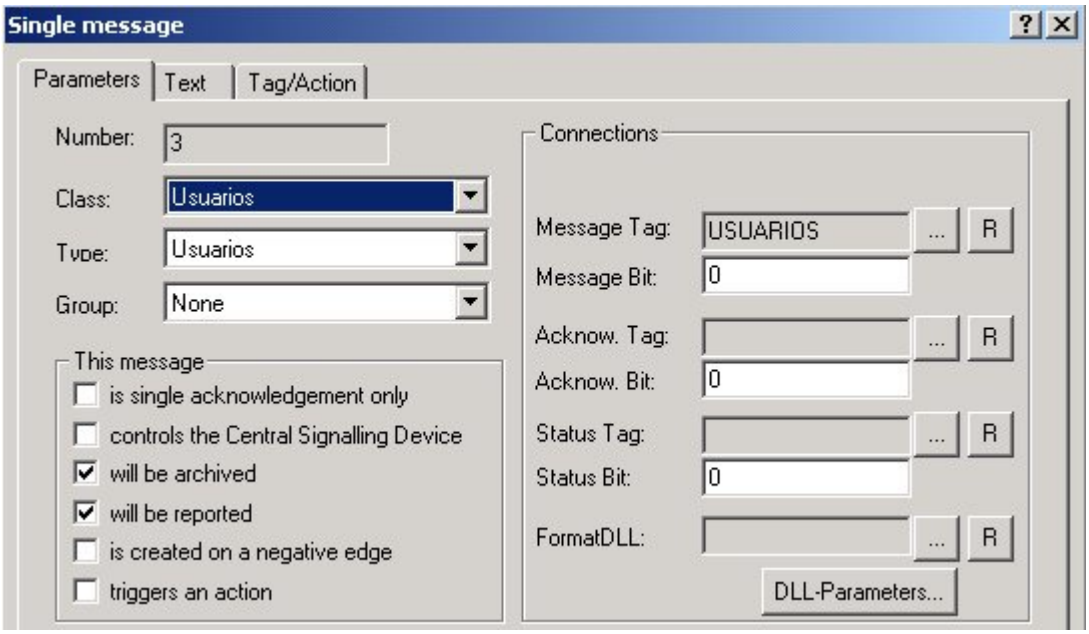
En la message classes usuarios, en sus propiedades, definimos que no tenga recomocimiento.



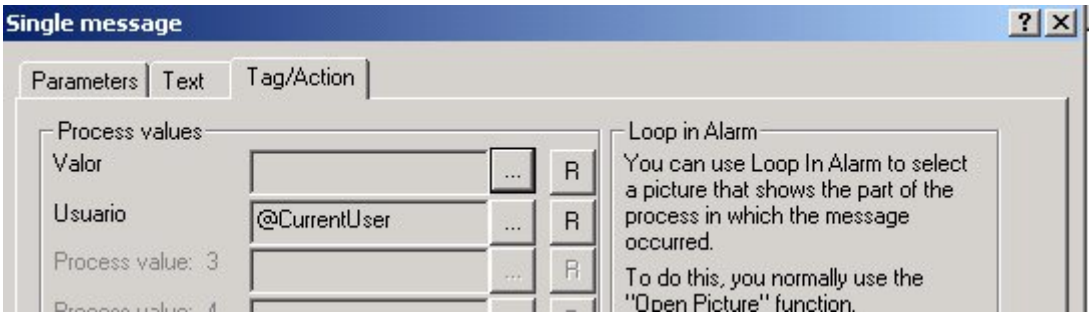
A continuación, en status text definimos los textos para identificar si se activa un usuario o se desactiva.



Lo siguiente es crear una nueva alarma, que será del siguiente tipo: Observar como hemos asociado la alarma a la variable USUARIOS creada anteriormente.



Esa línea de alarmas deberá contener el nombre del usuario que está realizando la acción, ya sea activarse o desactivarse. Esto lo conseguimos asociando al bloque de proceso usuario el nombre del usuario actual.



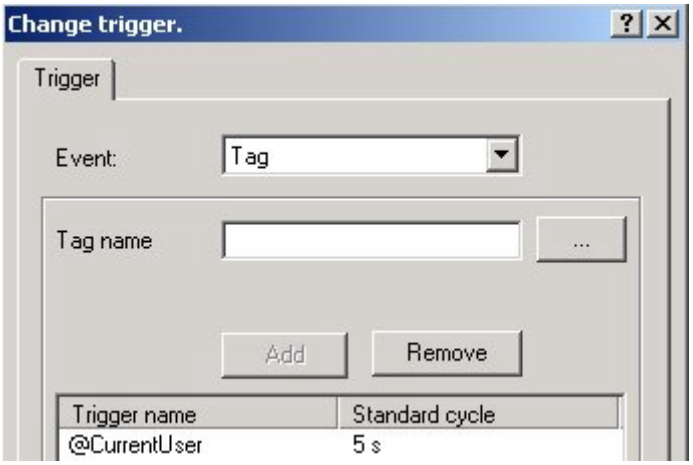
Ya hemos terminado en alarmas. Grabamos y salimos.

Ahora nos vamos al global script, y creamos una nueva actino, dentro de global actions, que llamaremos USUARIOS.PAS, y que contendrá el siguiente código.

```
int pepe;

pepe=strcmp(GetTagCharWait("@CurrentUser"),"");

if (pepe)
   SetTagBitWait("USUARIOS",TRUE);//Return - Type :BOOL
else
  SetTagBitWait("USUARIOS",FALSE);//Return - Type :BOOL

return 0;
```

El trigger  será el siguiente:

El problema viene ahora: si lo dejamos así, tal cual, no paarecerá el nombre de usuario dentro de la alarma. Esto se debe a que cuando se dispara el evento aún no le ha dado tiempo al WinCC a actualizar el valor de @currentuser en la base de datos, y aunque en memoria cambia y dispara esta subrutina, la alarma toma el valor de @currentuser no de la variable en memoria, sino de la base de datos, que aún no se ha actualizado. Es por esto que deberemos de retrasar el evento en el tiempo, para que la subrutina se ejecute una vez el valor de @currentuser esté alamacenado. Yo creo que con 5 segundos es suficiente.



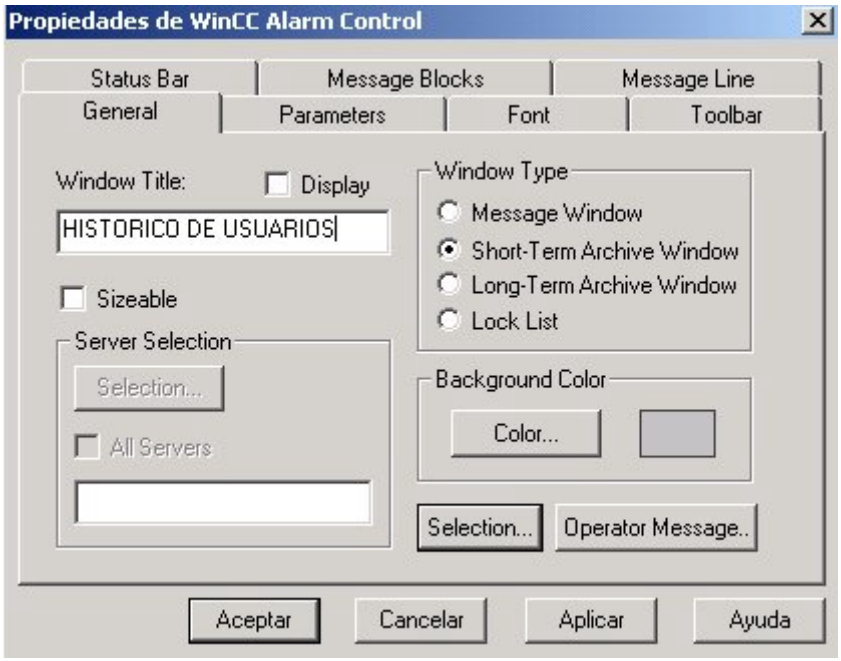
Ya tenemos la función global script, que recapitulando nos hace:

- Se dispara pasados 5 segundos desde que cambia el valor de @currentuser. Esta variable la cambia el sistema cuando nos logamos o deslogamos.
- Tenemos que identificar si pasa una cosa u otra, y de eso se encarga el código de la actino, activándonos o desactivándonos la variable que va asociada a la alarma que vimos antes.

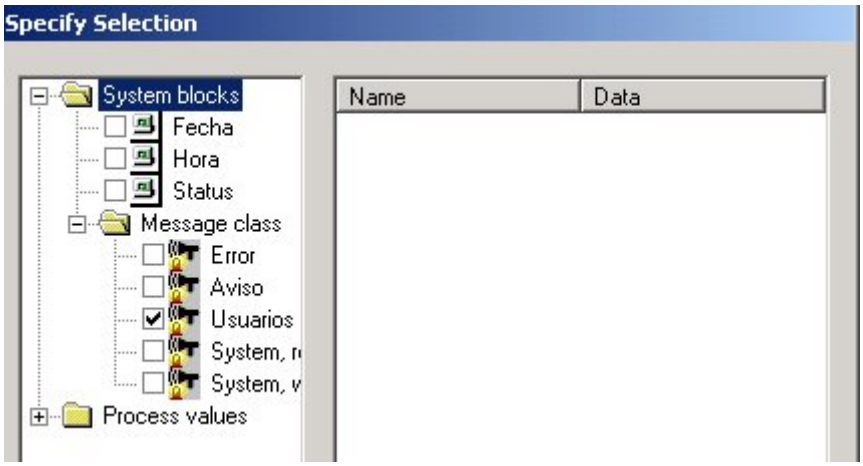
Con esto hemos acabado la parte del global. Ya puedes guardar y cerrar (recuerda que para que se ejecute el actino debe de estar activada la parte de runtime de global script.

Ahora nos abrimos la plantilla.pdl y la guardamos bajo el nombre historico_usuarios.pdl. Nos colocamos en bottom.pdl el correspondiente botón, como estamos haciendo, para poder saltar a la misma.

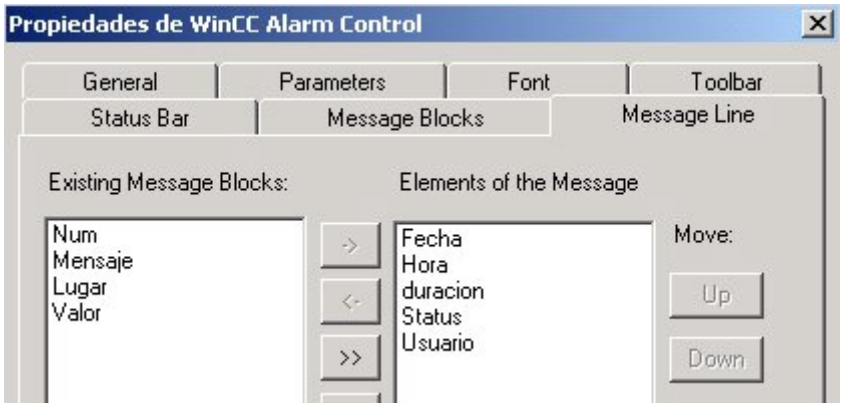
Nos colocamos en la misma un control de WinCC Alarm Control, y veamos que propiedades le asignamos (pinchando dos veces sobre el nuevo objeto).



Como se aprecia, existe un botón que pone selection. Vamos a seleccionar, pero ¿para qué?. Sencillo. Ten en cuenta que hemos hecho alarmas, y ahora una especial que es usuario. Si no le decimos a esta ventana que únicamente queremos visualizar en ella los usuarios nos va a mezclar en la ventana las alarmas con los usuarios, ya que para ella (y en realidad) todo son alarmas. Luego vamos a "filtrar" la ventana para que solo muestre nuestra única alarma llamada usuarios.

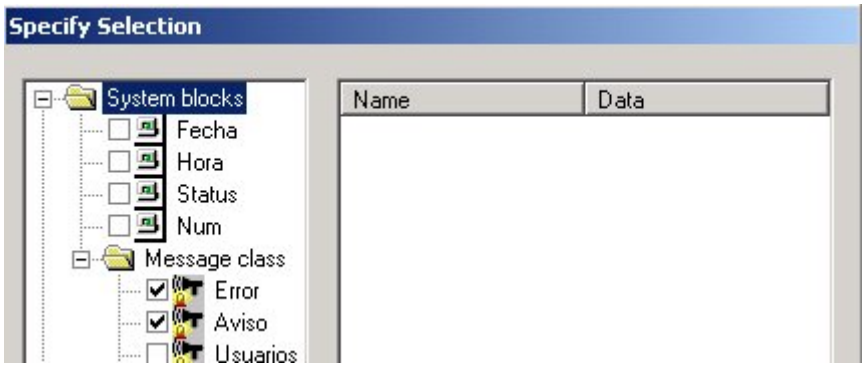


Tampoco queremos que nos muestre todos los bloques de que se compone una alarma, ya que el que más nos interesa es precisamente el del nombre de usuario, que justo es el que no aparecerá en las alarmas.



Bien, ya está. Si lo ejecutas comprobarás que funcionar, lo que se dice funcionar, funciona. Pero, mira debajo de la pantalla: ¡leches!, sale el usuario en la ventanita de bottom.pdl. Lógico. Para las demás ventanas de alarmas (la de bottom, y la que hicimos en el histórico) hemos hecho una nueva alarma, sin más. Tendremos pues que "filtrarla", pero esta vez en sentido excluyente.

Nos iremos pues a la picture bottom.pdl y realizaremos la siguiente selection (la misma operación será para la ventana del histórico de alarmas)..



El resultado final del histórico de usuarios debe ser algo parecido a esto:

	Fecha	Hora	duracion	Status	Usuario
15	17/09/01	22:53:23	0 00:03:23	DESACTIV	
16	17/09/01	22:56:45	0 00:00:00	ACTIVADO	
17	17/09/01	22:58:35	0 00:00:00	ACTIVADO	
18	17/09/01	22:58:48	0 00:00:12	DESACTIV	
19	17/09/01	22:59:08	0 00:00:00	ACTIVADO	
20	17/09/01	22:59:49	0 00:00:41	DESACTIV	
21	17/09/01	23:03:02	0 00:00:00	ACTIVADO	
22	17/09/01	23:03:12	0 00:00:10	DESACTIV	
23	17/09/01	23:05:07	0 00:00:00	ACTIVADO	administrator
24	17/09/01	23:07:30	0 00:02:22	DESACTIV	administrator
25	17/09/01	23:08:05	0 00:00:00	ACTIVADO	
26	17/09/01	23:08:07	0 00:00:02	DESACTIV	
27	17/09/01	23:08:14	0 00:00:00	ACTIVADO	
28	17/09/01	23:09:21	0 00:00:00	ACTIVADO	administrator
29	17/09/01	23:09:26	0 00:00:05	DESACTIV	
30	17/09/01	23:09:36	0 00:00:00	ACTIVADO	administrator
31	17/09/01	23:09:41	0 00:00:05	DESACTIV	
32	17/09/01	23:11:01	0 00:00:00	ACTIVADO	
33	18/09/01	00:05:42	0 00:00:00	ACTIVADO	
34	18/09/01	00:05:47	0 00:00:04	DESACTIV	
35	18/09/01	00:07:06	0 00:00:00	ACTIVADO	administrator
36	18/09/01	00:07:11	0 00:00:05	DESACTIV	
37	18/09/01	00:08:06	0 00:00:00	ACTIVADO	administrator
▶ 38	18/09/01	00:08:11	0 00:00:05	DESACTIV	

Date cuenta que cuando se desactiva un usuario el campo usuario de la línea de alarmas no contiene su nombre. Se puede dejar así (ya que está claro que se desloga el que antes se había logado, pues dos usuarios no pueden estar al mismo tiempo) o hacer una pequeña modificación al código que hemos visto antes utilizando una variable interna, pero no vale la pena complicarse la vida.

Prepara un report para este histórico, y verás que risa el día que digan que nadie ha tocado...jeje.