Integración mensajería instantánea con la Gestión de Incidencias

CONENIDOS

1. INTRODUCCIÓN

Las herramientas de gestión de incidencias permiten registrar los problemas de los usuarios y gestionar estos problemas asignando los tickets a los departamentos involucrados en su resolución. Permiten realizar otras muchas cosas y muy interesantes, pero ésta es la esencial. El alta de los tickets se debería realizar por un primer nivel de soporte, donde se tomen los datos más relevantes del problema, se registre el TICKET, y se asigne posteriormente al departamento apropiado.

En departamentos de TI pequeños, es muy importante intentar automatizar el alta de TICKETs, para mantener a las personas del departamento ocupadas en lo que realmente es su tarea: **intentar evitar que ocurran problemas y si ocurren, resolverlos**.

Desde el punto de vista del usuario, lo ideal es poder entablar una conversación de primera mano con el técnico encargado de la resolución, pero esto es algo prácticamente inasumible. Tendríamos a los técnicos todo el día hablando por teléfono, y los problemas sin resolver. En cuanto un usuario le contara su problema, y el técnico se dispusiera a investigar, resolver y perfeccionar, inmediatamente sería interrumpido por otro usuario interesado en explicar su problema por teléfono.

Pero al usuario hay que facilitarle la vida, no puede ser que para dar de alta una incidencia, tenga que rellenar un formulario con infinidad de campos, o campos que siempre son los mismos. No puede ser que para entrar en ese formulario se tarde más que en establecer una llamada de teléfono, porque entonces vas a tener la llamada.

La idea que se ha llevado a cabo es la siguiente: integrar la mensajería instantánea con la herramienta de gestión de Tickets. El objetivo que es el usuario pueda poner un Ticket sobre la marcha, y que sea más fácil que llamar por teléfono. La mensajería es ideal, porque: ¿quién no habla con el compañero de al lado por Messenger, a pesar de tener teléfono o la vía directa?Partimos del siguiente escenario:

- Herramienta de gestión de incidencias: Servicios de Soporte (Desarrollo en Delphi, contra base de Datos Oracle)
- Mensajería Instantánea Corporativa: Servidor Openfire y Clientes Pandion

Queremos llegar a lo siguiente:

Que el usuario establezca una conversación con un contacto (SAT) de su rooster de Pandion, escriba "Ticket: Descripción del problema" y automáticamente se genere un ticket a nombre de ese usuario en estado "Abierto".

Para ello el usuario SAT debe ser capaz de "hablar" con el resto de usuarios, y además debe ser capaz de lanzar comandos o scripts ante determinadas órdenes. El usuario virtual SAT debería usar un cliente de mensajería capaz de implementar esto, o dicho de otro modo, de funcionar en modo "robot"

El cliente que buscamos se llama **freetalk**.

2. INSTALACION DE FREETALK

La instalación de freetalk se realiza sobre Linux, en nuestro caso lo vamos a hacer sobre un CentOS. Es muy sencillo si tenemos algo de soltura en entornos UNIX. Nos ayudaremos de la aplicación `yum` para descargar e instalar los paquetes necesarios.

Para saber qué paquetes necesitamos, no hay más que leer el fichero INSTALL que bien con `freetalk`:

```
Dependencies
============

Loudmouth 1.x, GNU Guile, GNU readline, Glib 2.0
  gmessage (optional), xmessage (optional), dict (optional), beep (optional)
```

con `yum search` y una buena conexión a internet buscamos los paquetes que necesitamos, y con `yum install` los instalamos.

Para las aplicaciones que no encontremos los paquetes, habrá que conseguir el código fuente y compilar… para eso no hace falta sacar los apuntes de la universidad, tan solo leer atentamente el fichero INSTALL, y si sale algún error en el proceso no ignorarlo e investigar un poco hasta resolverlo.

`Loudmouth 1.4.3` y `freetalk 3.2` se han instalado a partir del código fuente siguiendo estos pasos:

- `./configure`
- `make`
- `make install`

Con esto ya podemos ejecutar `freetalk`; si nos da algún error quejándose de que no encuentra las librerías de `loudmouth`, quizás tengamos que modificar el fichero `ld.so.conf` para incluir el directorio con las librerías de `loudmouth: /usr/local/lib`

3. PRUEBAS FUNCIONALIDAD:

Antes de seguir, probaremos un poco el cliente de mensajería, para ver que se conecta sin problemas a nuestro servidor Openfire, si lo podemos lanzar en backgroud, si puede funcionar como robot, si es posible aceptar de forma automática las peticiones de subscripción de presencia, etc.

Nos olvidamos del funcionamiento en modo interactivo. Para configurar las opciones de conexión, deberemos modificar el fichero `freetalk.scm`, que estará en el directorio `.freetalk` de nuestro `HOME`.

```
;
;
;
        (ft-set-jid! "sat@contoso.es")
        (ft-set-password! "********")
        (ft-set-sslconn! #f)
        (ft-set-server! "wf.contoso.es")
        (ft-set-port! 52225)
        ;(ft-set-proxyserver! "proxyserver")
        ;(ft-set-proxyport! "52225")

 (add-hook! ft-login-hook
            (lambda (status)
              (if status
                  (begin
                    (ft-set-prompt! "~\\/~ ")
                    (ft-set-status-msg! "online Servicios Asistencia")
                    (ft-set-jid! "sat@contoso.es")
                                                                    ))))
```

En http://narnia.cs.ttu.edu/drupal/node/182 podemos ver un ejemplo de cómo interprear conversaciones y lanzar comandos, asi que de momento no hacemos muchas pruebas sobre esto.

Una consa importante es ver si cuando un usuario nos añade a su lista de contactos, la petición de subscripción que llega a `freetalk` se puede aceptar de forma automática. Investigando un pcoo vemos que modificando la extensión `rooster.scm` en el directorio `/usr/local/share/freetalk/extensions/`

```
(define (subscribe-recv jid)
  (ft-display (string-append (_ "[Buddy request recieved from ") jid (_ " use
/allow or /deny]")))
;;  (ft-display (string-append (_ "[Use /add ") jid (_ " to add him/her to
your buddy list]")))
;; Agregamos al usuario a nuestro rooster. Auto aceptamos la peticion
      ;;(ft-add-buddy (sans-surrounding-whitespace jid))
      (ft-subscription-allow (sans-surrounding-whitespace jid))
)
```

4. INSTALACIÓN ORACLE

Si queremos dar de alta un ticket en la aplicación de gestión de incidencias, tendremos que construirnos una "API" que sea capaz de hacer esto, y la mejor manera es llamar a un procedimiento almacenado de Oracle (PL/SQL). Como `freetalk` no puede llamar directamente a Oracle, nos serviremos de un script en KSH, al que pasaremos los argumentos necesarios. Este script será el encargado de ejecutar, mediante `sqlplus`, el procedimiento almacenado de Oracle.

Para disponer de `sqlplus` solamente tendremos que descargar estos paquetes de la WEB de Oracle e instalarlos:

```
rpm -ivh oracle-instantclient-basic-10.2.0.4-1.i386.rpm
rpm -ivh oracle-instantclient-sqlplus-10.2.0.4-1.i386.rpm
```

5. PRUEBA SQLPLUS

Antes de seguir, vamos a probar que `sqlplus` está correctamente instalado:

```
export LD_LIBRARY_PATH=/usr/lib/oracle/10.2.0.4/client/lib:/usr/lib
[root@srvv-zenoss ~]# mkdir /zenoss/oracle
```

Copiar aqui `tnsnames.ora` (fichero con los "connect strings" de Oracle, o los descriptores de las conexiones a nuestras bases de datos)

```
[root@srvv-zenoss ~]# export TNS_ADMIN=/zenoss/oracle
[root@srvv-zenoss ~]# sqlplus serviciossoporte@contoso
```

6. ALTA DE TICKET VIA SQL PLUS

Lo haremos a través de una función Oracle creada por radiga:

```
function INSERTA_TICKET(
                                    p_LOGIN_ALTA varchar2,
                                    p_LOGIN_AFECTADO varchar2,
                                    P_ID_PRIORIDAD_USUARIO NUMBER,
                                    P_ID_CLASIFICACION NUMBER,
                                    P_ID_TIPO_ORIGEN NUMBER,
                                    P_ASUNTO VARCHAR2,
                                    P_ID_AREA NUMBER,
                                    P_DEBE_NOTIFICARSE varchar
                                    ) return number is

 sqlInsert varchar2(4000);
 sqlinsert2  varchar2(4000);

 ID_TICKET number;
 ID_USU_ALTA number;
 ID_USU_AFEC number;
 ID_ESTADO number;
 ID_GRUPO number;
 ID_LOCALIZACION number;

begin

-- calcular id_ticket.
  EXECUTE IMMEDIATE 'SELECT MAX(ID_TICKET)+1 ID_TICKET FROM
SERVICIOSSOPORTE.TICKET' INTO ID_TICKET;

-- calcular el id_usuario alta.
  EXECUTE IMMEDIATE 'SELECT ID_USUARIO FROM LIBRA.BEG_USUARIOS_GRUPO WHERE
LOGIN ='|| '''' ||upper(P_LOGIN_ALTA)||'''' INTO ID_USU_ALTA;

-- calcular el id_usuario afectado.
  EXECUTE IMMEDIATE 'SELECT ID_USUARIO FROM LIBRA.BEG_USUARIOS_GRUPO WHERE
LOGIN ='|| '''' ||upper(P_LOGIN_AFECTADO)||'''' INTO ID_USU_AFEC;

-- calcular el id_localizacion del usuario afectado.
  EXECUTE IMMEDIATE 'SELECT ID_LOCALIZACION FROM LIBRA.BEG_USUARIOS_GRUPO
WHERE ID_USUARIO =' || ID_USU_AFEC INTO ID_LOCALIZACION;

-- grupo
  EXECUTE IMMEDIATE ' select id_grupo as GRUPO from
serviciossoporte.area_grupo where nivel_soporte=1 and id_area = 1' INTO
ID_GRUPO;

  ID_ESTADO := 0;

  sqlInsert := ' INSERT INTO serviciossoporte.ticket(id_ticket,
                                                     id_usuario_alta,
                                                     id_usuario_afectado,
                                                     Asunto,
```

```
                                                id_prioridad_usuario,
                                                id_estado,
                                                fecha,
                                                id_clasificacion,
                                                id_area,
                                                debe_notificarse,
                                                id_grupo,
                                                id_tipo_origen,
                                                id_localizacion,
                                                fecha_ocurrencia)';
   sqlinsert2 :=               ' values( '||ID_TICKET||','
                                       || ID_USU_ALTA ||','
                                       || Id_USU_AFEC ||','
                                       || ''''||P_ASUNTO||'''' ||','
                                       || P_ID_PRIORIDAD_USUARIO ||','
                                       || ID_ESTADO ||','
                                       || 'sysdate' ||','
                                       || P_ID_CLASIFICACION ||','
                                       || P_ID_AREA ||','
                                       || ''''||P_DEBE_NOTIFICARSE||''''
||','
                                       || ID_GRUPO ||','
                                       || P_ID_TIPO_ORIGEN ||','
                                       || ID_LOCALIZACION ||','
                                       || 'sysdate' ||')';

  sqlinsert := sqlinsert ||' '|| sqlinsert2;

  EXECUTE IMMEDIATE sqlInsert;

  EXECUTE IMMEDIATE 'commit';

  RETURN(ID_TICKET);

 EXCEPTION
   WHEN OTHERS THEN  RETURN -1; ROLLBACK;

end INSERTA_TICKET;
```

Parámetros:

- Login usuario alta
- Login usuario afectado
- Id_prioridad_usuario (1,2,3,4)
- Id_clasificacion (1->Fallo, 2->Consulta)
- Id_tipo_origen (4->Pandion)
- Asunto (2000)
- Id_area (Marca el área o departamento de la empresa al que va dirigido el ticket, en este caso usaremos 1 que indica Informática)
- Debe_notificarse (SI/NO Indica si se le informa al usuario por email de los cambios de estado)

A esta función la podemos llamar desde `sqlplus` fácilmente con la siguiente sintaxis:

```
SQL> variable ticket number;
```

```
SQL> exec :ticket :=
inserta_ticket('$us_alta','$us_afectado',id_prioridad,id_clasificacion,id_tip
o_origen,'Asunto',id_area,'SI');
SQL> print ticket
```

Y a `sqlplus` lo podemos llamar desde cualquier sitio, por ejemplo desde un script en `ksh`

7. MONITORIZACION FREETALK

Como `freetalk` se cuelga y se desconecta cada 5 minutos, y no hemos conseguido averiguar por qué, ni solucionarlo de ninguna manera, hemos hecho la ñapa el día, que incluso viene bien para que nuestro amigo virtual nunca se quede desconectado y nos deje vendidos ante nuestros usuarios.

Se trata de un script que se ejecuta cada 5 minutos, revisa si existe algún proceso de `freetalk` ejecutándose, en caso afirmativo mata estos procesos y lanza de nuevo la aplicación, con fuerzas renovadas para seguir dando de alta tickets por doquier.

```
#!/bin/ksh
#FECHA: 15/10/2009
#VERSION:1.1
#AUTOR: jjalonso
#NOMBRE: FreetalkMonitor
#FUNCION: Monitorizar que Freetalk se esa ejecutando
#VARIABLES

HISTORY_PATH=/root/.freetalk/history
HIST_MAX_SIZE=102400
WORK_DIR=/root
LOGFILE=freetalk_monitor.log


#ARGUMENTOS:

#FUNCIONES
usage ()
{
     echo "FreetalkMonitor"
     echo "uso: FreetalkMonitor "
}
#test if we have two arguments on the command line
if [ $# != 0 ]
then
    usage
    exit
fi
#Si hay alguna instancia de freetalk ejecutandose, la matamos
#Kill freetalk processes
echo "Revisando si hay procesos de freetalk ejecutandose..." 2>&1 >>
$WORK_DIR/$LOGFILE
for PID in `ps -fea | grep freetalk | grep -v grep | awk '{printf
"%s\n",$2}'`
do
        echo $PID
        kill -9 $PID 2>&1 >> $WORK_DIR/$LOGFILE
done
#Comprobamos espacio ocupado por logs
#Test space used by logs
echo "Comprobando si el espacio utilizado por los ficheros historicos supera
$HIST_MAX_SIZE..."  2>&1 >> $WORK_DIR/$LOGFILE
USED=`du -sk $HISTORY_PATH | /usr/bin/awk '{printf "%s\n",$1}'`
```

```
if [ $USED -ge $HIST_MAX_SIZE ]
then
        echo "Espacio en historico de freetalk superado" 2>&1 >>
$WORK_DIR/$LOGFILE
        #echo "Espacio en disco superado"
        rm -rf $HISTORY_PATH/* 2>&1 >> $WORK_DIR/$LOGFILE
else
        echo "Espacio en disco no superado" 2>&1 >> $WORK_DIR/$LOGFILE
        #echo "Espacio no superado"
fi


#Lanzamos Freetalk
#Run Freetalk
echo "Lanzando una nueva instancia de Freetalk..." 2>&1 >> $WORK_DIR/$LOGFILE
su - root -c /usr/local/bin/freetalk 2>&1 >> $WORK_DIR/$LOGFILE
```

El problema de matar la sesión de Freetalk y volver a lanzarla cada 5 minutos, es que los usuarios que tengan activadas las notificaciones de presencia, van a recibir notificaciones cada 5 minutos.

Para ello, hemos modificado el código del cliente Pandion para que si la notificación de presencia bien del usuario "SATUR - Servicio de Asistencia Técnica URgente", no se emita ninguna notificación.

El fichero que hemos cambiado se llama ClientRooster.js, y estas son las líneas:

```
//if ( Presence.Type == 'available' && external.globals( 'cfg' )(
'alertonline' ).toString() == 'true' && ! this.Items( ShortAddress
).Resources.Count && external.globals( 'connecttime' ) + 30000 < ( new Date()
).getTime() )
//no recibir alertas si el usuario que se conecta es SAT@contosso.es
if ( ShortAddress  != 'sat@contosso.es' && Presence.Type == 'available' &&
external.globals( 'cfg' )( 'alertonline' ).toString() == 'true' && !
this.Items( ShortAddress ).Resources.Count && external.globals( 'connecttime'
) + 30000 < ( new Date() ).getTime() )
                                                {
```

8. LLAMADA A CREATICKET.KSH

La idea es llamar a un script ksh y pasar argumentos como el usuario que nos envía el Ticket, y el asunto del Ticket. Además habrá que responder al usuario con lo que devuelva el script en ksh, en teoría el número de Ticket creado. Y lo primero, para que el usuario sepa que puede crear Tickets hablando con freetalk, habrá que informarle de alguna manera cuando el usuario se dirija a freetalk: "¿Hola, tu quien eres?".

Para ello nos creamos una extensión, ticket.scm, que se guarda en el directorio $HOME/.freetalk/extensions:

```
;;Extension inspirada en Ejemplo visto en
http://narnia.cs.ttu.edu/drupal/node/182
(define (ticket timestamp from nickname msg)
"Forward command"
(if (ignored-message? msg)
(ft-hook-return)
(cond ((string-prefix-ci? "TICKET" msg)
(begin
(ft-display (string-append "Se ha llamado a /root/CreaTicket.ksh
sat@contosso.es " from " 2 1 4 NO
 '" msg "'"))
(ft-send-message-no-hook from  "No te preocupes, voy a dar de alta un ticket
en tu nombre y nos
 pondremos con ello en cuanto podamos.")
(sleep (+ 1 (random 3 (seed->random-state (current-time)))))
(send-message-pipe from (string-append "/root/CreaTicket.ksh sat@contosso.es
" from " 2 1 4 NO '"
msg "'"))
(sleep (+ 1 (random 3 (seed->random-state (current-time)))))
(ft-send-message-no-hook from  "Recuerda que puedes consultar el estado de
tus tickets en http:
//app.contosso.es/GestionServicios/")
(sleep (+ 1 (random 3 (seed->random-state (current-time)))))
(ft-send-message-no-hook from  "Muchas gracias por ayudarnos a ser mas
eficientes")
(ft-hook-return)
))
((string-prefix-ci? "Gracias" msg)
(begin
(sleep (+ 1 (random 3 (seed->random-state (current-time)))))
(ft-send-message-no-hook from  "De nada")
(ft-hook-return)
))
((string-prefix-ci? "Hola" msg)
(begin
(sleep (+ 1 (random 3 (seed->random-state (current-time)))))
(ft-send-message-no-hook from  "Hola, soy un asistente virtual y ayudo al
Dpto. de Informatica
```

```
con el alta de Tickets. Si tienes un problema, por favor escribe TICKET
seguido de un espacio y
 una descripcion de lo que te pasa. En cuanto podamos te llamaremos.")
(ft-hook-return)
))
((string-prefix-ci? "Adios" msg)
(begin
(sleep (+ 3 (random 3 (seed->random-state (current-time)))))
(ft-send-message-no-hook from  "Hasta Luego")
(ft-hook-return)
))
((string-prefix-ci? "de nada" msg)
(begin
(sleep (+ 3 (random 3 (seed->random-state (current-time)))))
(ft-send-message-no-hook from  "Hasta Luego")
(ft-hook-return)
))
)
)
)
(add-hook! ft-message-receive-hook ticket)
```

Esta extensión se carga desde el script `freetalk.scm` comentado anteriormente:

```
;
;
        (ft-set-jid! "sat@contoso.es")
        (ft-set-password! "Vepatv100")
        (ft-set-sslconn! #f)
        (ft-set-server! "wf.contoso.es")
        (ft-set-port! 52225)
        ;(ft-set-proxyserver! "proxyserver")
        ;(ft-set-proxyport! "52225")

 (add-hook! ft-login-hook
            (lambda (status)
              (if status
                  (begin
                    (ft-set-prompt! "~\\/~ ")
                    (ft-set-status-msg! "online Servicios Asistencia")
                    (ft-set-jid! "sat@contoso.es")
                                                                        ))))
;;(add-hook! ft-disconnect-hook (lambda (reason) (ft-connect) (ft-display
"Adios")))
(ft-load "ticket.scm")
;;;
;;; Let ctrl-a display full roster, ctrl-e who i am, ctrl-h online buddies
 (ft-bind-to-ctrl-key #\a "(/who \"all\")")
 (ft-bind-to-ctrl-key #\e "(/whoami \"\")")
 (ft-bind-to-ctrl-key #\h "(/who \"\")")
;;;
;;;
```

Y por último, aquí tenemos el script CreaTicket.ksh que genera un ticket con los argumentos que se le pasan, llamando a la función SQL que hemos visto anteriormente.

```
# #!/bin/ksh
#FECHA: 15/10/2009
#VERSION:1.1
#AUTOR: jjalonso
#NOMBRE: CreaTicket
#FUNCION: Crear Un Ticket en Servicios de Soporte desde ksh
#VARIABLES de ENTORNO
WORK_DIR=/root
LOGFILE=crea_ticket.log


#export NLS_LANG=SPANISH_SPAIN.WE8ISO8859P
export LD_LIBRARY_PATH=/usr/lib/oracle/10.2.0.4/client/lib:/usr/lib
export TNS_ADMIN=/zenoss/oracle

#ARGUMENTOS:
#.      Login usuario alta
#.      Login usuario afectado
#.      Id_prioridad_usuario (1,2,3,4)
#.      Id_clasificacion (1->Fallo, 2->Consulta)
#.      Id_tipo_origen (4->Pandion)
#.      Debe_Notificar (SI,NO)
#.      Asunto (2000)
#.ORDEN de los parametros a pasar a la funcion de SQL:

#inserta_ticket(p_login_alta => :p_login_alta,
#                           p_login_afectado => :p_login_afectado,
#                           p_id_prioridad_usuario =>
:p_id_prioridad_usuario,
#                           p_id_clasificacion => :p_id_clasificacion,
#                           p_id_tipo_origen => :p_id_tipo_origen,
#                           p_asunto => :p_asunto,
#                           p_id_area => :p_id_area,
#                           p_debe_notificarse => :p_debe_notificarse);

#FUNCIONES
usage ()
{
     echo "CreaTicket"
     echo "uso: CreaTicket UsuarioAlta UsuarioAfectado IDPrioridad
ID_Clasificacion, ID_tipo_origen Debe_Notificar Asunto"
}
#test if we have two arguments on the command line
if [ $# != 7 ]
then
    usage
    exit
fi
#Chequeo de la informacion
echo "*****************************************************************" >>
$WORK_DIR/$LOGFILE
date >> $WORK_DIR/$LOGFILE
us_alta=`echo $1 | awk -F@ '{printf "%s\n",$1}'`
us_afectado=`echo $2 | awk -F@ '{printf "%s\n",$1}'`
echo "Se va a crear un Ticket con la siguiente informacion:" 2>&1 >>
$WORK_DIR/$LOGFILE
```

```
echo "* Usuario alta: $1" 2>&1 >> $WORK_DIR/$LOGFILE
echo "* Usuario afectado: $2" 2>&1 >> $WORK_DIR/$LOGFILE
echo "* ID Prioridad: $3" 2>&1 >> $WORK_DIR/$LOGFILE
echo "* ID_Clasificacion: $4" 2>&1 >> $WORK_DIR/$LOGFILE
echo "* ID_tipo_origen: $5" 2>&1 >> $WORK_DIR/$LOGFILE
echo "* Debe_Notificar: $6" 2>&1 >> $WORK_DIR/$LOGFILE
echo "* Asunto: $7" 2>&1 >> $WORK_DIR/$LOGFILE
echo "Ticket Creado"
#Llamamos a SQLPlus
#En Produccion:
#sqlplus serviciossoporte/ss@CONTOSSO << EOF
ID_TICKET=`sqlplus -s serviciossoporte/ss@CONTOSSO << EOF
variable ticket number;
exec :ticket :=
inserta_ticket('$us_alta','$us_afectado',$3,$4,$5,'$7',1,'$6');
print ticket;
exit;
EOF`
echo "Su numero de Ticket es el siguiente:"
echo $ID_TICKET | awk -FT '{printf "%s\n",$3}'
```

9. CONCLUSIONES

Hemos agilizado un proceso clave dentro de la gestión de incidencias, como es el alta de Tickets

Hemos generado un valor para el negocio, ya que somos capaces de soportar más usuarios sin incrementar el personal de TI, ni bajar el nivel de servicio.

Y todo ello sin gastar un duro del presupuesto de TI, ya que Openfire, Freetalk y Pandion son gratis.