“SAPping”

Notas técnicas de SAP - Tip en detalle Nro. 06

(Lo nuevo, lo escondido, o simplemente lo de siempre pero bien explicado)

---

*"Tips en breve/Tips en detalle" se envía con frecuencia variable y absolutamente **sin cargo** como un servicio a nuestros clientes SAP. Contiene principalmente notas técnicas y no contiene mensajes publicitarios.*

Conteste este mail con asunto “REMOVER” si no desea recibir más esta publicación. Si desea suscribir otra dirección de e-mail para que comience a recibir los tips envíe un mensaje desde esa dirección a sapping@teknoda.com

Uso del “Logging” de aplicaciones en actualizaciones a la base de Datos, hechas con CALL TRANSACTION

Tema: Application Log, CALL TRANSACTION, Base de Datos.

Utilidad: Application Log para monitorear actualizaciones a la base de datos hechas con CALL TRANSACTION.

Nivel: Intermedio.

Introducción

Cuando se realizan actualizaciones a la base de datos, el sistema registra internamente información sobre el resultado de cada grabación (si fue exitosa o no), estadísticas (cantidad de registros ingresados o actualizados, cantidad de registros erróneos, etc), además de datos específicos de la aplicación (código de material ingresado o actualizado, por ejemplo).

Cuando las actualizaciones se llevan a cabo a través del CALL TRANSACTION, esta información **NO se hace disponible al usuario en forma automática,** sino que, requiere de sentencias adicionales para que **quede almacenada** en una **tabla interna de mensajes.** Luego, debe **analizarse y procesarse también desde la programación**, para decodificar el formato de la tabla interna en información entendible para el usuario. Para lograr este objetivo, existen las funciones de “application logging” (registro de mensajes). Desde ya, el “Logging” de aplicaciones como concepto, puede implementarse también en otro tipo de programas.

Mediante la técnica desarrollada en este tip, que discute el caso del CALL TRANSACTION, los logs quedarán asociados a un objeto y “disponibles” para el usuario, para ser consultados en cualquier momento. Se podrá seleccionar por objeto, fecha, hora, usuario, transacción, nivel de importancia del log, etc, y los mensajes se presentarán en forma amigable a través de semáforos que indican el éxito o fracaso de cada grabación, botones para ordenar, visualizar en detalle, dezplazar la pantalla, seleccionar, etc.

Pasos a seguir para utilizar esta técnica:

Creación del objeto en la tablas de log de aplicación: Antes de acceder al código fuente del programa es necesario **crear un objeto** en la tabla de objetos del log de aplicación. Para ello, con la transacción SM30, se da de alta una entrada en la tabla BALOBJ, indicando nombre y texto descriptivo. Dicho objeto se utilizará como parámetro principal de los módulos de función invocados.

Dentro del fuente del programa se necesita:

1.Definir las variables para el manejo de los mensajes y los parámetros de las funciones. Ejemplo:

```
data: appl_log_obj like balhdr-object value ’ZOBJ’. "Objeto del log
data: appl_log_class_high like balmi-probclass value ’1’. "Clase de problema
data: appl_log_class_info like balmi-probclass value ’4’. "Clase de problema
data: lognumber like balnri occurs 0 with header line. "Nros de logs
data: i_msg like balmi.      "Estructura de mensajes y logs(técnicos)
data: i_msg_bis like balmi.  "Estructura de mensajes y logs(de aplicación)
data: msg_error type c value ‘E’.   "Tipo de mensaje (Error)
data: msg_abort type c value ‘A’.   "Tipo de mensaje (Cancelar)
data: msg_warning type c value ‘W’. "Tipo de mensaje (Aviso)
data: msg_info type c value ‘I’.    "Tipo de mensaje (Informativo)
data: msg_ok type c value ‘S’.      "Tipo de mensaje (Exito)
data: rc like sy-subrc.          "Código de retorno del Call Transaction
```

2. Definir la tabla interna que contendrá los mensajes de la transacción. Ejemplo:

```
data:    begin of mess_tab occurs 0.
           include structure bdcmsgcoll.
data:    end   of mess_tab.
```

3. Inicializar el log:

Esta función se invoca por única vez, al iniciar el ciclo que llena la tabla BDC. Verifica que el objeto o subobjeto exista y borra todos los datos existentes asociados:

```
   ...
   call function 'APPL_LOG_INIT'
       exporting
            object                = appl_log_obj
*           subobject             = ´ ´
*           log_handle            = ´ ´
       exceptions
            object_not_found      = 1
            subobect_not_found    = 2
            others                = 3.
   ...
```

4. Grabar el log:

Se completan las estructuras que se pasan como parámetros de la función, recorriendo la tabla interna de mensajes . También pueden armarse mensajes propios con datos específicos de la aplicación.

Esta función se invoca después de ejecutar el comando CALL TRANSACTION; se graba un mensaje y si no existe ninguna entrada para ese objeto, ésta es creada. Si no se especifica objeto o subobjeto, se asume el que se usó más recientemente. Ejemplo:

```
...
call transaction ´Mxxx´ using bdc_tab mode ´N´ update ´S´ messages
into mess_tab.

rc = sy-subrc.

perform grabar_log using rc {zvar1} {zvar2}.
...

form grabar_log using status {zvar1} {zvar2}.

  clear: i_msg, i_msg_bis.
  if status ne 0.
* Mensaje de error:
    loop at mess_tab.
      move-corresponding mess_tab to i_msg.
      i_msg-msgno     = mess_tab-msgnr.
      i_msg-msgty     = msg_error.
      i_msg-probclass = appl_log_class_high.
    endloop.

* Se completa el segundo renglón del log con datos específicos de
la aplicación:

    i_msg_bis-msgty     = msg_error.
    i_msg_bis-probclass = appl_log_class_high.
    i_msg_bis-msgid     = `{Zidmens1}`.
    i_msg_bis-msgno     = `999`.
    i_msg_bis-msgv1     = {zvar1}.
    i_msg_bis-msgv2     = {zvar2}.

  else.

* Mensaje de éxito:
    i_msg-msgty     = msg_ok.
    i_msg-msgid     = `{Zidmens2}`.
    i_msg-msgno     = `nnn`.
    i_msg-msgv1     = {zvar1}.
    i_msg-msgv2     = {zvar2}.

  endif.
```

```
   call function 'APPL_LOG_WRITE_SINGLE_MESSAGE'
       exporting
            object                = appl_log_obj
*           subobject             = ´ ´
*           log_handle            = ´ ´
            message               = i_msg
            update_or_insert      = ´I´
       exceptions
            object_not_found      = 1
            subobect_not_found    = 2
            others                = 3.
   if sy-subrc <> 0.
     ...
   elseif not I_msg_bis is initial.

     call function 'APPL_LOG_WRITE_SINGLE_MESSAGE'
         exporting
              object                = appl_log_obj
*             subobject             = ´ ´
*             log_handle            = ´ ´
              message               = i_msg_bis
              update_or_insert      = ´I´
         exceptions
              object_not_found      = 1
              subobect_not_found    = 2
              others                = 3.
     if sy-subrc <> 0.
     endif.

   endif.


 endform.
```

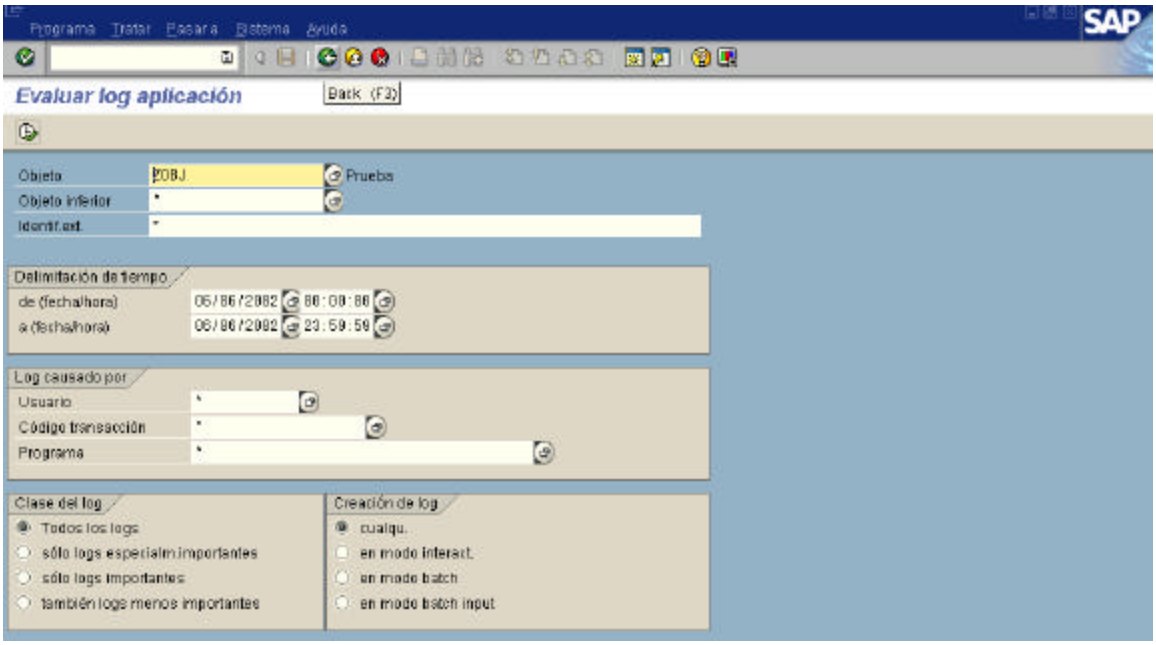
5. Finalizar el log:

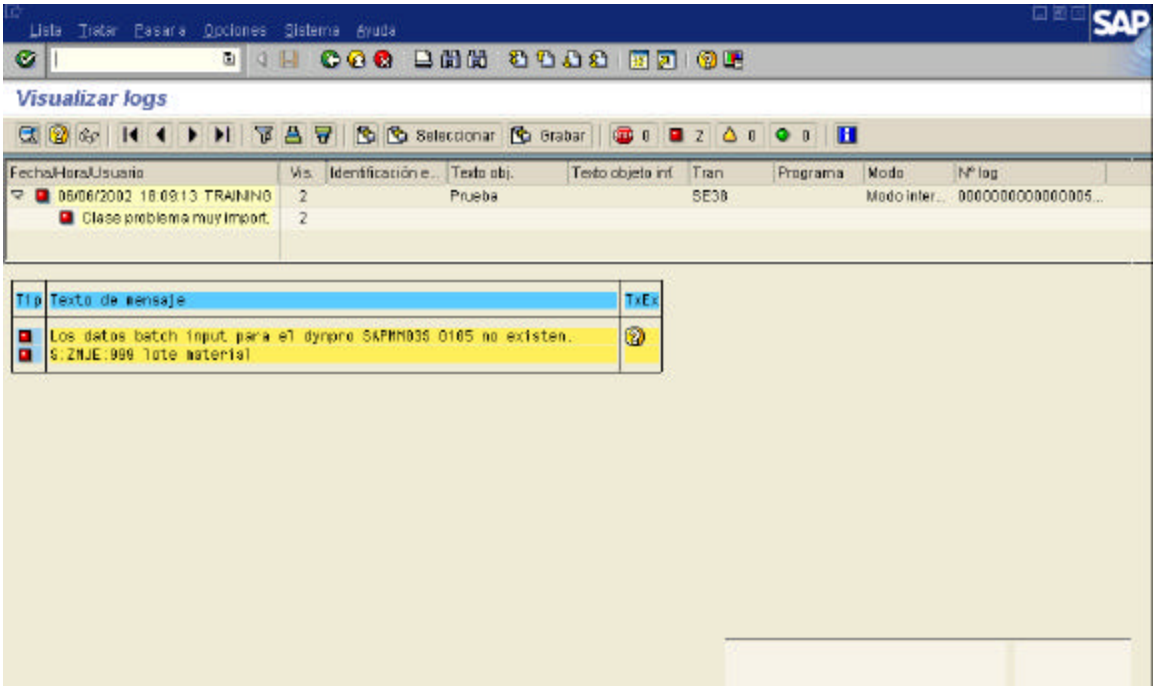
Esta función se invoca por única vez, al terminar el ciclo que llena la tabla BDC. Graba todos los datos del objeto o subobjeto y si el log es nuevo, el número de log es devuelto al programa llamador:

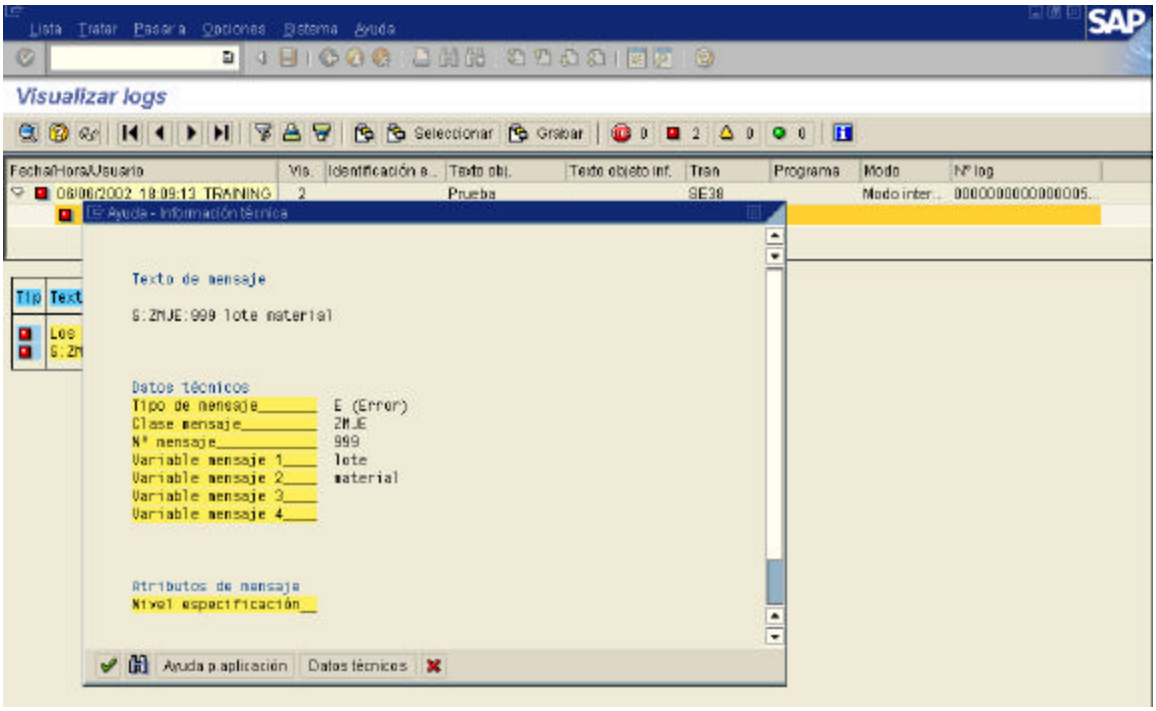
```
    ...
    refresh lognumber.
    call function 'APPL_LOG_WRITE_DB'
        exporting
             object                = appl_log_obj
 *           subobject             = ´ ´
        tables
             object_with_lognumber  = lognumber
         exceptions
             object_not_found      = 1
             subobject_not_found   = 2
             internal_error        = 3
             others                = 4.
    ...
```

6. Visualizar el log:

Luego de correr el programa, se ejecuta la transacción SLG1. (Evaluar log de aplicación):







Para tener en cuenta:

Existen otras formas de generar este tipo de log, como así también otras funciones relacionadas. Por ejemplo, se puede grabar más de un mensaje con el módulo de función APPL_LOG_WRITE_MESSAGES.

IMPORTANTE

*Dudas o consultas: **sapping@teknoda.com***