Fábrica - Modelos Logs - Log de Registro de Errores

Este es el log utilizado para registrar los errores generados en los procesos de las aplicaciones de la compañia.

¿Donde usar el Servicio?

Este servicio debe consumirse en las generaciones de errores de:

- Aplicaciones Java
- Aplicaciones .Net
- Web Service
- Integración Supertabla
- Soluciones que afecten los productos SICOF

Nota: Aplicaciones Powerbuilder

Powerbuilder tiene restricciones para el consumo de servicios Rest por lo tanto en las aplicaciones de esta tecnología se implementará un API para realizar las llamadas.

Diccionario de Datos

OWNER	SICOF	**TABLE**	MOV_LOG_ERRORES	**COMMENTS**	Contiene el registro de errores genrados en los procesos de las aplicaciones SICOF
#	**NAME**	**NULLEABLE**	**TYPE**	**COMMENTS**	**WS**[1)]
1	CODIGO_INTERNO	N	NUMBER(38,0)	Identificador interno del registro (autoincrementado por secuencia)	Interno, Autoincremental
2	COD_ORA_ERROR	Y	NUMBER	Código del error (Puede usuarse para errores del manejador de base de datos pero el campo es multiproposito)	Externo, si es un error de base de datos se debe incluir el código del error

3	MSJ_ORA_ERROR	Y	VARCHAR2(1024)	Mensaje simple del error (Resumen)	Externo, si es un error de base de datos se debe incluir el mensaje del error
4	NOM_PROCEDIMIENTO	Y	VARCHAR2(128)	Nombre del método, procedimiento, clase o contexto de ejecución local donde se genera el error	Externo, Requerido
5	FECHA	Y	DATE	Fecha en la cuál se genera el error	No utilizado
6	USUARIO_BD	Y	VARCHAR2(32)	Usuario de conexión de la base de datos	Externo, Requerido
7	OBSERVACION	Y	VARCHAR2(2048)	Mensaje extendido del error (Detalle)	Externo, Requerido
8	HOST_CLIENTE	Y	VARCHAR2(50)	Host del cliente (Dirección IP)	Externo, Requerido
9	FECHA_REGISTRO	Y	DATE	Fecha del sistema DB	Interno, Formato dd/mm/yyyy hh:mm:ss, Requerido
10	CODIGO_USUARIO	Y	NUMBER	Código del usuario de la sesión en la cuál se genera el error	Externo, Requerido
11	CODIGO_MEMPRESA	Y	VARCHAR2(64)	Código de la empresa de la sesión en la cuál se genera el error	Externo, Requerido
12	CODIGO_APLICACION	Y	NUMBER	Código de la aplicación (Identificador interno numérico)	Externo, Requerido
13	INFO_APP	Y	VARCHAR2(256)	Información de la aplicación (En las situaciónes donde no se identifique código interno se puede enviar el nombre de la aplicación o información adicional)	Externo
14	SESSION_MAC	Y	VARCHAR2(64)	MAC del equipo del usuario	Externo
15	SESSION_BROWSERVERSION	Y	VARCHAR2(64)	Versión del Navegador	Externo, Requerido
16	SESSION_OSTYPE	Y	VARCHAR2(64)	Sistema Operativo	Externo, Requerido

17	ERROR_LOG	Y	CLOB	Utilizada solo para casos especiales donde sea necesaria registrar información amplia del error	Externo, Requerido
18	ERROR_TYPE	Y	VARCHAR2(64)	Tipo del error (LEVE, MODERADO, CRITICO por defecto)	Externo

Columna: WS

Se adiciona esta columna para identificar reglas asociadas a la implementación de los servicios web que permiten gestionar el almacenamiento de los logs. La columna es una referencia y no hace parte del servicio sin embargo las reglas que se definen en ella si aplican para la columna relacionada:

Reglas

- **Interno**: Indica que el campo se gestiona dentro del servicio y por lo tanto no se pedira en los parametros.
- **Autoincremental**: Indica que el campo se comporta como una secuencia.
- **Externo**: Indica que el campo debe estar en los parametros del consumo.
- **Requerido**: Indica que el campo debe ser enviado en el consumo y el servicio debe validarlo para continuar.
- **Obsoleto**: Indica que el campo ya no es utilizado en la nueva implementación.
- **No utilizado**: Indica que el campo no será utilizado en ninguna implementación.

Nota

- Todas las operaciones del servicio que gestiona la persistencia de la tabla deben estar documentadas incluyendo la definición de los campos, formatos, longitudes de columnas e indicar si es requerido o no.

Columna: ERROR_LOG

Esta columna sirve para almacenar información extra del error generado[2)] que se generar en los procesos. Se define la siguiente estructura base ejemplo:

```
{
  "errors":[
    {
      "error_code":"Requerido: Número del error o nombre de la excepción",
      "error_description": "Requerido: Descripción técnica del error",
      "error_tracer": "Pila de mensajes capturada en el error"
      "error_line": "Linea del proceso donde se genera el error"
      "error_method": "Método donde se genera el error"
```

```
    }
  ]
}
```

Donde:

- **errors**: Array de errores en el proceso
- **error_code**: Propiedad contenida en cada indice del array json. Representa el código del error, alguna aplicaciones pueden represnetar este código como un número o como nombre de excepción.
- **error_description**: Propiedad contenida en cada indice del array json. Representa el mensaje técnico del error generado.
- **error_tracer**: Propiedad contenida en cada indice del array json. Representa la traza del error generado.
- **error_line**: Propiedad contenida en cada indice del array json. Representa la linea donde se ha generado el error.
- **error_method**: Propiedad contenida en cada indice del array json. Representa el método donde se ha generado el error.

Ejemplo:

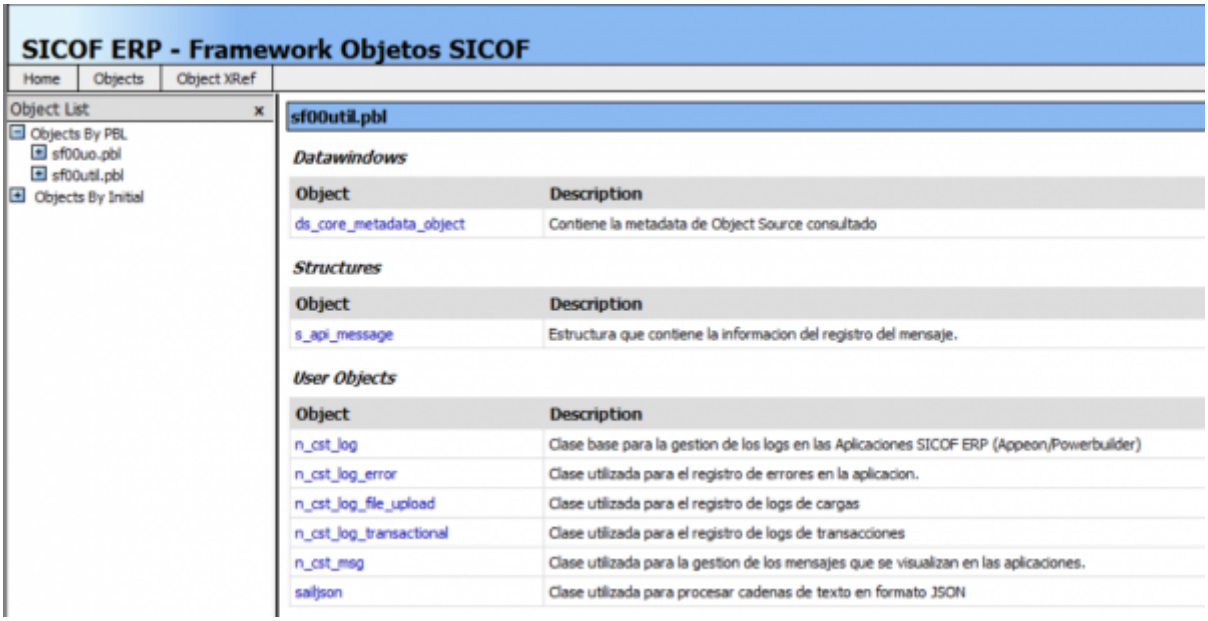
```
{
  "errors":[
    {
      "error_code":"2501",
      "error_description": "Error actualizando saldos de rubros.",
      "error_tracer": "ORA-0001 PK Constrainst Invalid"
      "error_line": "25"
      "error_method": "of_calcular_saldos"
    },
    {
      "error_code":"java.lang.NullPointerException",
      "error_description": "Error actualizando saldos de rubros.",
      "error_tracer": "Exception in thread main
java.lang.NullPointerException
    at Printer.printString(Printer.java:13)
    at Printer.print(Printer.java:9)
    at Printer.main(Printer.java:19)"
      "error_line": "23"
      "error_method": "of_calcular_saldos"
    }
  ]
}
```

Notas

- Los valores de las columnas deben ser registrados como String

Modo de uso: Powerbuilder - Documentación

Para visualizar la documentación debe descargar el siguiente repositorio Documentación, abrir la pagina Index.html en su navegador web la cual es similar a la siguiente imagen:



En ella encontrará la documentación de las librerias que hacen parte del framework **Objetos SICOF** el cuál se irá actualizando frecuentmente a medida que se documenten las clases.

La Libreria que contiene la funcionalidad de los logs es la librería **sf00util.pbl**

Los Objetos relacionados en el API son:

- **n_cst_app**: Clase contenedora de objetos logs
- **n_cst_log_errores**: Clase para la gestión de errores

Ejemplos de Uso

Para facilitar la implementación y uso del API de gestión de errores se crea un objeto interno privado en la clase global **guo_app** el cual puede ser accedido por el método **of_log_error()** que devuelve la instancia del objeto. Sin embargo para implementaciones específicas se puede optar por crear y administrar la clase de error **n_cst_log_error** según considere el desarrollador.

A continuación se listan ejemplos de uso el cuál presenta las forma de utilizar el API, para más información debe consultar la documentación en el repositorio.

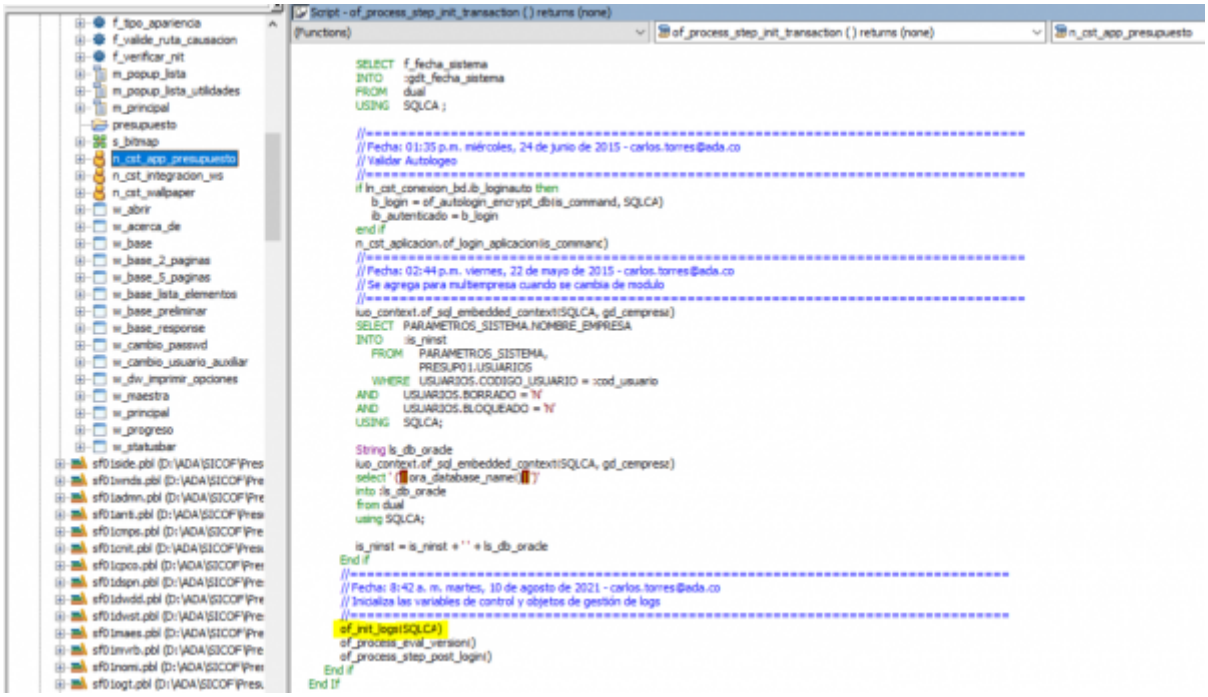
```
/*Ejemplos de uso utilizando la instancia genérica de la clase guo_app*/
guo_app.of_log_error( ).of_add_log("EJEMPLO_CODE", 10, true, SQLCA)
guo_app.of_log_error( ).of_add_log("EJEMPLO_CODE", "ERROR_DB", true, SQLCA)
guo_app.of_log_error( ).of_add_log("EJEMPLO_CODE", 10, ls_args, true, SQLCA)
guo_app.of_log_error( ).of_add_log_text("Ejemplo", "ERROR_DB",
"Presupuesto", true, SQLCA)
guo_app.of_log_error( ).of_add_log_text( sqldbcode, sqlerrtext, dataobject,
```

```
sqlsyntax, ls_ventana, true, lts_db)

/*Ejemplo de uso definiendo la clase de error,
instancia el array de formateo de mensaje y
registra en el log el mensaje formateado.
Posteriormente elimina la instancia de la clase de gestión de error.
"El array de formateo se utiliza cuando el mensaje se genera por
expresiones:
Ejemplo:
Mensaje Base = Hola #1 (identificado con el código de mensaje SALUDO_CODE),
se crea un array con un expresión de la siguiente forma
array[1] = "Mundo",
de esta forma al usar el método of_add_log se genera el mensaje: Hola
Mundo".
*/
n_cst_log_error luo_log_error
String ls_args[]
ls_args[1] = "Mundo"
luo_log_error = Create n_cst_log_error
luo_log_error.of_add_log("SALUDO_CODE", 10, ls_args, true, SQLCA)
destroy luo_log_error

/*Ejemplo de uso definiendo la clase de error y
registra en el log el mensaje por medio de un código de mensaje.
Posteriormente elimina la instancia de la clase de gestión de error.*/
n_cst_log_error luo_log_error
luo_log_error = Create n_cst_log_error
luo_log_error.of_add_log("EJEMPLO_CODE", 10, true, SQLCA)
destroy luo_log_error

/*Ejemplo de uso definiendo la clase de error y
registra en el log el mensaje por medio de un código de mensaje.
Posteriormente elimina la instancia de la clase de gestión de error.*/
n_cst_log_error luo_log_error
luo_log_error = Create n_cst_log_error
luo_log_error.of_add_log("EJEMPLO_CODE", "ERROR_DB", true, SQLCA)
destroy luo_log_error
```

Consideraciones

- El API puede ser activada o desactivada por medio de la constante: **LOG_ERRORS** (Solo en aplicaciones SICOF ERP (Appeon/Powerbuilder)) siempre y cuando se utilice la implementación de la clase **guo_app**.
- El desarrollador es el encargado de gestionar la transacción que realiza la persistencia.
- Se automatiza el registro de errores en los objetos **uo_datawindow** y **uo_datastore** en toda su herencia.
- Se automatiza el registro de errores genéricos no controlados lanzados por el evento **systemerror** de Powerbuilder.

- Para el procesamiento de logs de bloques se implementa Clase sailjson para procesamiento de cadenas de texto en ese formato.
- Cada módulo (Contabilidad, Presupuesto, Tesorería, Compras, Talento y Nómina) debe implementar el método de inicialización **guo_app.of_init_logs(SQLCA)** en el método **of_process_step_init_transaction** de la clase **guo_app** especializada por cada módulo. A continuación se muestra una imagen de referencia de la implementación del módulo de presupuesto. Utilice esta guía para implementaciones en otros módulos teniendo presente que la clase **n_cst_app** se especializa con el nombre de la aplicación que la contiene. Ejemplo: en presupuesto la clase especializada es **n_cst_app_presupuesto**, por lo general la clase esta en la libreria principal que contiene el objeto **Application**.



API Json

Se adicionan métodos de procesamiento de bloques en formato json los cuales pueden ser utilizados para registrar trazas de error a continuación se muestran ejemplos de uso del API.

```
sailjson ljson, ljson1
String ls_content

ljson = create sailjson
ljson.setattribute( 'version', '1001')
//add json object
ljson1 = ljson.addobject( 'header')
ljson1.setattribute( 'count', 3)
ljson1.setattribute( 'comment', 'items count')

//add json object array, first item
ljson1 = ljson.addarrayitem( 'data')
ljson1.setattribute( 'colid', 1)
ljson1.setattribute( 'colname', 'aaaaaa')
ljson1.setattribute( 'coladdr', '')
```

```
//add second item of the array
ljson1 = ljson.addarrayitem( 'data')
ljson1.setattribute( 'colid', 2)
ljson1.setattribute( 'colname', 'bbbbbbbb')
setnull(ls)
ljson1.setattribute( 'coladdr', ls)
//add third item of the array
ljson1 = ljson.addarrayitem( 'data')
ljson1.setattribute( 'colid', 3)
ljson1.setattribute( 'colname', 'cccccc')

ljson.setattribute( 'creattime', string(now(), 'yyyymmdd.hhmmss'))

ls_content =  ljson.getformatjson('' )
destroy ljson
```

Limitaciones

Se identifica que actualmente la versión de Appeon (2013/2016) presenta problemas con el procesamiento del API Json para estructuras complejas como Arrays por esta razon cada procesamiento del API debe ser realizado a un solo nivel de complejidad. A continuacion se muestra un ejemplo de uso funcional en Appeon/Powerbuilder que puede ser tomado como referencia.

```
/*PASO 1: Definir un objeto de la clase sailjson en la zona de instancia del componente donde se va utilizar el API como una ventana, objeto no visual, etc*/
sailjson iuo_json
```

```
/*PASO 2: Incializar el objeto iuo_json antes de utilizarlo*/
iuo_json= create sailjson
```

```
/*
PASO 3: No es obligatorio, pero se recomienda crear un método local de asignación para simplificar el proceso como ejemplo se define metodo of_add_log_error con los siguientes argumentos
Integer ai_indx
String as_error_code
String as_error_description
String as_error_line
String as_error_method
String as_error_tracer
*/

//Validaciones previas de control
if IsNull(ai_indx) then ai_indx = 0
if IsNull(as_error_code) then as_error_code = ""
if IsNull(as_error_description) then as_error_description = ""
if IsNull(as_error_line) then as_error_line = ""
```

```
if IsNull(as_error_method) then as_error_method = ""
if IsNull(as_error_tracer) then as_error_tracer= ""

//Registrar las propiedades del error con el indice
iuo_json.setattribute("error" + String(ai_indx) + "_code", as_error_code)
iuo_json.setattribute("error" + String(ai_indx) + "_description",
as_error_description)
iuo_json.setattribute("error" + String(ai_indx) + "_line", as_error_line)
iuo_json.setattribute("error" + String(ai_indx) + "_method",
as_error_method)
iuo_json.setattribute("error" + String(ai_indx) + "_tracer",
as_error_tracer)
```

```
//En el metodo donde se gestiona el error se consume el API Log de Errores
con traza en formato json, se recomienda uitlizar una transacción diferente
a la del proceso para que se pueda almacenar el error
guo_app.of_log_error( ).of_add_log("EJEMPLO_CODE", 10, lson, true, SQLCA)

//Por último se debe reiniciar el componente para otro proceso
iuo_json = Create sailjson
```

```
//Invocar el método local of_add_log_error
of_add_log_error(1, "-1", 'Error actualizando base de datos', '100',
'SystemError', '')
```

Modo de uso: Java

Para las aplicaciones desarrolladas en las tecnologías (Web):

- Java
- .Net
- PHP

el log de sesión será implementado por medio de un Servicio Web el cual deberá considerar las reglas de Columna: WS

←Volver atras

[1)]
Define las reglas que debe aplicar el Web Service
[2)]
Trazas