# Implementando Transparent Application Failover sobre Aplicaciones .NET

Por Francisco Riccio

## Introducción

Oracle Database ha venido ofreciendo durante versiones anteriores su tecnología Oracle Real Application Cluster (RAC) como solución de Alta Disponibilidad de Base de Datos, el cual tiene una arquitectura similar a la que se presenta:

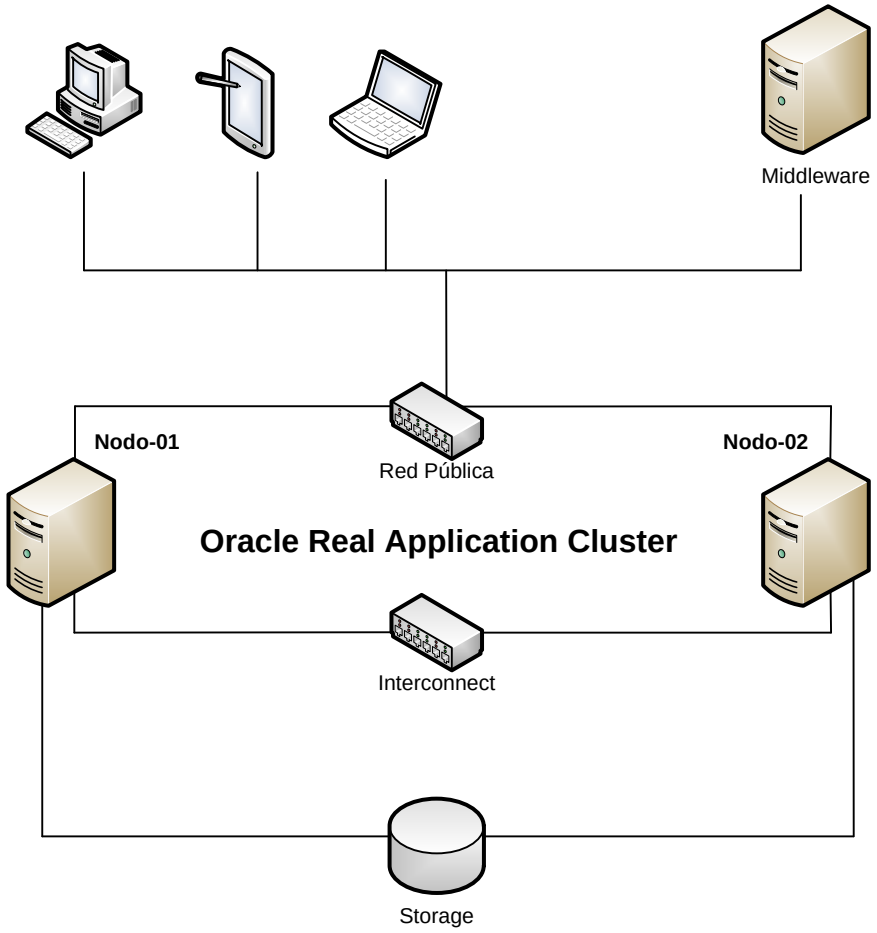



**Figura 1**

Donde diferentes aplicaciones (cliente/servidor o web) irán conectándose a los diferentes servidores de base de datos que conforman la solución de Oracle RAC.

Revisar el siguiente link para más información sobre Oracle Real Application Cluster:

http://www.oracle.com/us/products/database/options/real-application-clusters/overview/index.html

Muchas aplicaciones montadas sobre una arquitectura en Oracle RAC, suelen considerar que una conexión de base de datos establecida siempre estará disponible; pero en la realidad no siempre

será así; debido a que el servidor de base de datos puede tener un reinicio, congelamiento o cualquier incidente provocando que sus conexiones de base de datos se desconecten.

Por lo cual, surgen algunas preguntas como: ¿Nuestras aplicaciones están manejando el evento de pérdidas de conexiones?, ¿En caso de una desconexión, el usuario de la aplicación lo percibirá?, ¿En caso de una reconexión de base de datos que sucede con la información que estuvo manipulando el usuario?. Seguramente estas preguntas ya dejaron a más de uno preocupado.

Oracle Database nos ofrece 2 tecnologías disponibles que permiten que nuestras aplicaciones puedan contar con mecanismos de alta disponibilidad a nivel de sesión:

Estas 2 tecnologías son: Transparent Application Failover (TAF) y Fast Connection Failover (FCF). Ambos son excluyentes.

Más adelante se explicará en detalle TAF como objetivo de este documento.

Es importante mencionar que estás tecnologías se implementan mediante servicios de base de datos y no es exclusivo para Oracle RAC, el cual es un error conceptual bastante frecuente. Es decir, nosotros podemos implementar TAF y FCF en ambientes de base de datos single instance protegidos mediante Oracle Restart o también mediante Oracle Data Guard o cualquier tecnología de replicación que permita mantener una copia de la base de datos productiva donde las aplicaciones se reconectarán.

Recomiendo revisar la Arquitectura de Máxima Disponibilidad de Oracle (MAA) en el siguiente url:

http://www.oracle.com/technetwork/database/features/availability/maa-096107.html

A continuación se presentará un ejemplo didáctico de cómo implementar Aplicaciones .NET con TAF con la finalidad de contar con un mecanismo de alta disponibilidad no solo a nivel de base de datos sino también a nivel de conexión.

Se asume que el lector maneja ODP.NET, en caso no lo sea, recomiendo revisar el siguiente url:

http://www.oracle.com/technetwork/es/articles/dotnet/implementar-aplicaciones-net-1924201-esa.html

La implementación que más adelante se mostrará, se ha realizado sobre un Oracle RAC 12cR1 (12.1.0.1) en configuración Standard Cluster sobre plataforma Oracle Linux 5 Update 9 de 64 bits y .NET Framework 4.0 con ODP.NET 4. El IDE de desarrollo es sobre Visual Studio 2010 basado en el lenguaje C#.

## Implementación - Transparent Application Failover (TAF)

TAF es una característica ofrecida mediante el driver OCI (Oracle Call Interface) y permite automáticamente reconectar las conexiones fallidas de base de datos, es decir, ejecuta un failover de sesión de base de datos. Solo está disponible para aplicaciones que utilicen las librerías OCI y está disponible por defecto. TAF no está disponible para conexiones creadas por RMAN.

Es importante notar que una reconexión recién se iniciará cuando la aplicación intente ejecutar alguna operación sobre la base de datos usando su conexión fallida y no antes.

La reconexión realizada, generará una nueva conexión idéntica a la original pero con las siguientes observaciones:

- La reconexión automática utilizará la misma cadena de conexión que mantuvo la conexión fallida.
- Configuraciones específicas de una sesión tales como: Formato de Fecha, Territorio, Idioma, etc., no serán establecidas en la nueva conexión. Esto debe ser manejado mediante código en la aplicación como veremos más adelante.
- Si una desconexión de base de datos ocurre cuando se estuvo realizando una operación de FETCH mediante un cursor, este podrá ser continuado en la nueva conexión que se establezca, es decir, el cursor podrá seguir devolviendo las filas faltantes y permitiendo la continuidad del proceso funcional de la aplicación.
- Cualquier transacción que estuvo pendiente de confirmación se le ejecutará un rollback automáticamente durante el proceso de failover de sesión.
- Variables establecidas en código anónimo de PL/SQL o en paquetes no serán mantenidas en la nueva conexión establecida.

Existen múltiples propiedades que pueden ser configuradas para TAF, las cuales pueden ser realizadas desde el lado del servidor (configuración en el servicio de base de datos) o desde el lado de la aplicación (configuración en la cadena de conexión). En caso se realicen en ambos lados (servidor/cliente) siempre predomina las propiedades establecidas por el lado del servidor.

Recomiendo siempre establecer las propiedades de TAF en el servicio de base de datos porque de esta manera centralizamos la configuración en un solo lugar, en vez de preocuparnos de realizar la aplicación en cada estación cliente.

Las propiedades que pueden ser establecidas son:

| Parámetro | Descripción |
|---|---|
| METHOD | BASIC: Crea una conexión de base datos en el momento de solicitar un failover.<br><br>PRECONNECT: Se establece una conexión secundaria a la original, utilizando mínimos recursos para esta segunda conexión y estará preparada en caso de un failover, de manera que la reconexión sea más rápida. Esta opción solo está |

| | |
|---|---|
| | disponible cuando trabajamos con servicios de base de datos administrados por el administrador y debe contar con al menos 1 instancia disponible (AVAILABLE) para el servicio. |
| TYPE | NONE (default): Previene de ejecutar un failover de conexión.<br><br>SESSION: Permite la reconexión automática sin restablecer las operaciones de FETCH que los cursores estuvieron realizando antes de la operación de failover.<br><br>SELECT: Permite la reconexión automática y reestablece las operaciones de FETCH que estuvieron ejecutando los cursores antes del failover, es decir continua la consulta SQL que estuvo en progreso. Adicionalmente utiliza una cantidad de memoria adicional para mantener información necesaria para el failover, como: el row fetch último, el SCN utilizado en el momento de obtener los datos y el SQL Plan Hash.<br><br>Nota: Oracle Database ejecuta de nuevo la sentencia que estuvo ejecutándose antes del incidente en la nueva conexión realizada, utilizando el mismo SCN que fue utilizado en la sentencia original. |
| DELAY | Tiempo de espera en segundos para intentar una reconexión. |
| RETRIES | Número de intentos de reconexión que se realizarán hasta encontrar una conexión exitosa. |

**Configuración:**

a) Configurar la cadena de conexión hacia la base de datos.

Ejemplo utilizando VIP IP.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="CONEXION" value="Data Source=(DESCRIPTION = (ADDRESS_LIST = (ADDRESS =
(PROTOCOL = TCP)(HOST = 142.68.1.150)(PORT = 1521))(ADDRESS = (PROTOCOL = TCP)(HOST =
142.68.1.151)(PORT = 1521)))(CONNECT_DATA=(SERVICE_NAME = TAF_APP)(SERVER =
DEDICATED)));User Id=friccio;Password=oracle"/>
  </appSettings>
</configuration>
```

Ejemplo utilizando SCAN IP.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="CONEXION" value="Data Source=(DESCRIPTION = (ADDRESS_LIST = (ADDRESS =
(PROTOCOL = TCP)(HOST = srvrac-scan)(PORT = 1521)))(CONNECT_DATA=(SERVICE_NAME =
TAF_APP)(SERVER = DEDICATED)));User Id=friccio;Password=oracle "/>
  </appSettings>
</configuration>
```

ó

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="CONEXION" value="Data Source=TAF_APP;User Id=friccio;Password=oracle;"/>
  </appSettings>
</configuration>
```

De ambos ejemplos mostrados, se puede apreciar que la cadena de conexión está utilizando un servicio de base de datos llamado TAF_APP.

b) Configuración del servicio TAF.

Por un mayor entendimiento, el servicio lo crearé en tres partes.

Paso 1: Definiendo el servicio sin alguna propiedad específica.

```
[oracle@srvasm1 ~]$ srvctl add service -d PRD -s TAF_APP -preferred PRD1,PRD2
[oracle@srvasm1 ~]$
```

Paso 2: Modificando el servicio con TAF con Método BASIC y Tipo de Failover SELECT.

```
[oracle@srvasm1 ~]$ srvctl modify service -d PRD -s TAF_APP -P BASIC
[oracle@srvasm1 ~]$

[oracle@srvasm1 ~]$ srvctl modify service -d PRD -s TAF_APP -failovermethod BASIC
[oracle@srvasm1 ~]$

[oracle@srvasm1 ~]$ srvctl modify service -d PRD -s TAF_APP -failovertype SELECT
[oracle@srvasm1 ~]$
```

Paso 3: Configurando el número de reintentos a 6 y con una espera en cada reintento de 30 segundos.

```
[oracle@srvasm1 ~]$ srvctl modify service -d PRD -s TAF_APP -failoverdelay 30
[oracle@srvasm1 ~]$ srvctl modify service -d PRD -s TAF_APP -failoverretry 6
[oracle@srvasm1 ~]$
```

Validando el Servicio:

```
[oracle@srvasm1 ~]$ srvctl config service -d PRD -s TAF_APP
Service name: TAF_APP
Service is enabled
Server pool: PRD_TAF_APP
Cardinality: 2
Disconnect: false
Service role: PRIMARY
Management policy: AUTOMATIC
DTP transaction: false
AQ HA notifications: false
Global: false
Commit Outcome: false
Failover type: SELECT
Failover method: BASIC
TAF failover retries: 6
TAF failover delay: 30
Connection Load Balancing Goal: LONG
Runtime Load Balancing Goal: NONE
TAF policy specification: BASIC
Edition:
Pluggable database name:
Maximum lag time: ANY
SQL Translation Profile:
Retention: 86400 seconds
Replay Initiation Time: 300 seconds
Session State Consistency:
Preferred instances: PRD1,PRD2
Available instances:
```

**Nota 1:** A partir de la versión Oracle Database 11gR2, TAF puede trabajar con FAN con la finalidad de disminuir el tiempo de detección del servicio caído de base de datos y así iniciar el failover de sesión rápidamente.

Para habilitar FAN, debemos crear el servicio con la opción: -q true.

Más información sobre FAN revisar el siguiente URL:

http://docs.oracle.com/cd/E11882_01/java.112/e16548/apxracfan.htm#JJDBC28935

**Nota 2:** Cuando una base de datos no se encuentra en Oracle RAC ni tampoco está trabajando con Oracle Restart, los servicios que deseemos implementar deberán ser creados a través del paquete DBMS_SERVICE. Para un mayor detalle revisar el siguiente url:

http://docs.oracle.com/cd/E16655_01/appdev.121/e17602/d_serv.htm

**Nota 3:** Si deseamos crear un servicio con método PRECONNECT debemos crearlo de la siguiente manera:

```
[oracle@srvasm1 ~]$ srvctl add service -d PRD -s TAF_APP_PRECON -r PRD1 -a PRD2 -P PRECONNECT
[oracle@srvasm1 ~]$ srvctl start service -d PRD -s TAF_APP_PRECON
[oracle@srvasm1 ~]$ srvctl status service -d PRD -s TAF_APP_PRECON
Service TAF_APP_PRECON is running on instance(s) PRD1
```

En este caso se ha creado un servicio con una Instancia Preferida (PRD1) y una Disponible (PRD2).

El servicio no debe ser creado con los parámetros -failovermethod y -failovertype.

Una vez creado, verificamos que ahora existen 2 recursos nuevos creados en el OCR (crsctl stat res -t):

```
ora.prd.taf_app.svc
      1         ONLINE  ONLINE       srvasm1                  STABLE
      2         ONLINE  ONLINE       srvasm2                  STABLE
ora.prd.taf_app_precon.svc
      1         ONLINE  ONLINE       srvasm2                  STABLE
ora.prd.taf_app_precon_preconnect.svc
      1         ONLINE  ONLINE       srvasm1                  STABLE
```

Posterior a la creación del servicio, el cliente debe mantener una configuración especial en su cadena de conexión.

Ejemplo:

```
EJEM_TAF_APP_PRECON =
  (description =
   (address = (protocol=tcp)(host=srvrac-scan)(port=1521))
   (connect_data=
     (service_name=TAF_APP_PRECON)
     (failover_mode=
       (backup=EJEM_TAF_APP_PRECON_preconnect)
       (type=select)
       (method=preconnect)
     )
   )
  )

EJEM_TAF_APP_PRECON_preconnect =
  (description =
   (address = (protocol=tcp)(host=srvrac-scan)(port=1521))
   (connect_data=
     (service_name=TAF_APP_PRECON_preconnect)
   )
  )
```

Validamos la configuración de TAF de la sesión en la base de datos:

```
SQL> select FAILOVER_METHOD,FAILOVER_TYPE,FAILED_OVER from v$session where username='SYSTEM';

FAILOVER_M FAILOVER_TYPE FAI
---------- ------------- ---
PRECONNECT SELECT        YES
```

*.- El campo FAILED_OVER indica si ya hubo un failover de sesión ejecutado.

*.- Cuando verificamos en la vista V$SESSION o GV$SESSION los campos FAILOVER_METHOD y FAILOVER_TYPE podrían mostrarse con el valor de NULL si la sesión aún no ha ejecutado ningún script en la base de datos. Este comportamiento es esperado y está documentado en My Oracle

Support (MOS) Nota: 1490078.1 (FAILOVER_TYPE FAILOVER_METHOD shows NONE in v$session or gv$session even though failover (TAF) is configured).

c) Implementación del evento Failover (Opcional).

ODP.NET nos da la posibilidad de ejecutar un código .NET en nuestra aplicación al momento de ejecutarse un failover de sesión. Este evento le pertenece a la clase OracleConnection.

**Definiendo el objeto Conexión y el Evento Failover.**

```
private OracleConnection getConexion()
{
    string conexion = System.Configuration.ConfigurationManager.AppSettings["CONEXION"].ToString();
    if (_conexion == null)
    {
        _conexion = new OracleConnection(conexion);
        _conexion.Open();
        configurarPropiedades();
        reloj.Enabled = true;
        lblestado.Text = "Estado: Normal";
        _conexion.Failover += new OracleFailoverEventHandler(OnFailover);
    }
    return _conexion;
}
```

**Implementando el evento Failover.**

```
public FailoverReturnCode OnFailover(object sender, OracleFailoverEventArgs eventArgs)
{
    switch (eventArgs.FailoverEvent)
    {
        case FailoverEvent.Begin:
        {
            MessageBox.Show("Inicio el failover", "Failover", MessageBoxButtons.OK, MessageBoxIcon.Stop);
            break;
        }
        case FailoverEvent.Abort:
        {
            lblestado.Text="Estado: Failover fue cancelado";
            break;
        }
        case FailoverEvent.End:
        {
            configurarPropiedades();
            lblestado.Text = "Estado: Failover terminado";
            break;
        }
        case FailoverEvent.Error:
        {
            lblestado.Text = "Estado: Failover ha fallado pero se esta reintentando...";
            this.Refresh();
            return FailoverReturnCode.Retry;
        }
        case FailoverEvent.Reauth:
        {
            lblestado.Text = "Estado: Usuario re-autentificandose en la base de datos...";
            break;
        }
        default:
        {
            lblestado.Text = "Estado: Failover fallido";
            break;
        }
    }
    return FailoverReturnCode.Success;
}
```

Nota: Al finalizar el failover (FailoverEvent.End) es importante evidenciar como configuramos propiedades específicas de una conexión, ya que ellas no serán configuradas por TAF automáticamente como previamente se mencionó.

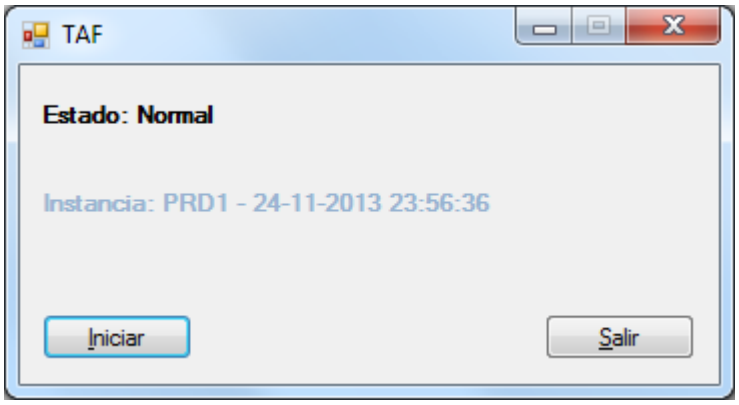
```
private void configurarPropiedades()
{
    OracleGlobalization globalizacion = _conexion.GetSessionInfo();
    globalizacion.Language = "AMERICAN";
    globalizacion.DateFormat = "Dd Month yyyy";
    _conexion.SetSessionInfo(globalizacion);
}
```

Para obtener todas las propiedades de personalización que pudieran realizarse, recomiendo revisar el siguiente url:

http://docs.oracle.com/cd/B19306_01/win.102/b14307/OracleGlobalizationClass.htm

**Realizando pruebas de failover de sesión.**

- Aplicación ejecutándose y conectada a la instancia a la instancia PRD1.




- Validamos en la base de datos que el usuario FRICCIO tiene configurado las propiedades de TAF habilitadas para su conexión.

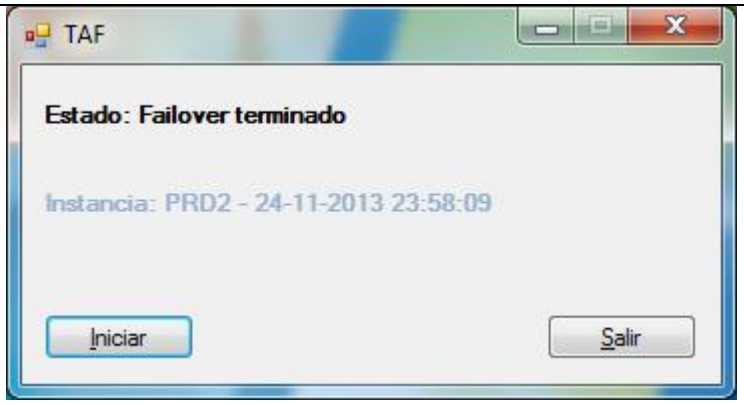
```
SQL> select FAILOVER_METHOD,FAILOVER_TYPE from v$session where username='FRICCIO';

FAILOVER_M FAILOVER_TYPE
---------- -------------
BASIC      SELECT
```

- Matamos el proceso SMON de la instancia PRD1 con la finalidad de provocar un failover de sesión.

```
[oracle@srvasm1 ~]$ ps -ef | grep smon
grid      4491     1  0 17:16 ?        00:00:00 asm_smon_+ASM1
root      4523     1  1 17:16 ?        00:06:11 /u01/app/12.1.0/grid/bin/osysmond.bin
oracle   28446     1  0 23:35 ?        00:00:00 ora_smon_PRD1
oracle   29158 19607  0 23:57 pts/2    00:00:00 grep smon
[oracle@srvasm1 ~]$ kill -9 28446
```

- Veremos que inicia el Failover.


- Finaliza el failover de sesión, donde ahora la aplicación está ejecutando sus requerimientos en la instancia PRD2.




Se detalla todo el código fuente del programa.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Oracle.DataAccess.Client;

namespace AppTAF
{
    public partial class FrmTAF : Form
    {

        private OracleConnection _conexion = null;

        public FrmTAF()
        {
            InitializeComponent();
        }

        private void configurarPropiedades()
        {
            OracleGlobalization globalizacion = _conexion.GetSessionInfo();
            globalizacion.Language = "AMERICAN";
            globalizacion.DateFormat = "Dd Month yyyy";
            _conexion.SetSessionInfo(globalizacion);
        }

        private OracleConnection getConexion()
        {
            string conexion =
System.Configuration.ConfigurationManager.AppSettings["CONEXION"].ToString();
            if (_conexion == null)
            {
                _conexion = new OracleConnection(conexion);
                _conexion.Open();
                configurarPropiedades();
                reloj.Enabled = true;
                lblestado.Text = "Estado: Normal";
                _conexion.Failover += new OracleFailoverEventHandler(OnFailover);
            }
            return _conexion;
        }
```

```
        private void reloj_Tick(object sender, EventArgs e)
        {
            try
            {
                OracleCommand cm = _conexion.CreateCommand();
                cm.CommandText = "select instance_name||' - '||to_char(sysdate,'DD-MM-YYYY
HH24:MI:SS') as texto from v$instance";
                cm.CommandType = CommandType.Text;
                string instancia = cm.ExecuteScalar().ToString();
                lblinstancia.Text = "Instancia: " + instancia;
            }
            catch (Exception e1)
            {
                Console.WriteLine(e1.Message);
            }
        }

        public FailoverReturnCode OnFailover(object sender, OracleFailoverEventArgs
eventArgs)
        {
            switch (eventArgs.FailoverEvent)
            {
                case FailoverEvent.Begin:
                {
                    MessageBox.Show("Inicio el failover", "Failover", MessageBoxButtons.OK,
MessageBoxIcon.Stop);
                    break;
                }
                case FailoverEvent.Abort:
                {
                    lblestado.Text="Estado: Failover fue cancelado";
                    break;
                }
                case FailoverEvent.End:
                {
                    configurarPropiedades();
                    lblestado.Text = "Estado: Failover terminado";
                    break;
                }
                case FailoverEvent.Error:
                {
                    lblestado.Text = "Estado: Failover ha fallado pero se esta
reintentando...";
                    this.Refresh();
                    return FailoverReturnCode.Retry;
                }
                case FailoverEvent.Reauth:
                {
                    lblestado.Text = "Estado: Usuario re-autentificandose en la base de
datos...";
                    break;
                }
                default:
                {
                    lblestado.Text = "Estado: Failover fallido";
                    break;
                }
            }
                  return FailoverReturnCode.Success;
        }

        private void btniniciar_Click(object sender, EventArgs e)
        {
            getConexion();
        }
```

```
        private void FrmTAF_Load(object sender, EventArgs e)
        {
            lblinstancia.Text = "";
            lblestado.Text = "";
        }

        private void btnsalir_Click(object sender, EventArgs e)
        {
            reloj.Enabled = false;
            if (_conexion != null)
            {
                _conexion.Close();
            }
            _conexion = null;
            Application.Exit();
        }

    }
}
```

## Conclusiones

Se puede apreciar que Oracle TAF ayuda a complementar nuestras soluciones de Alta Disponibilidad proporcionándonos un failover de sesiones a nuestras aplicaciones; ocasionando que nuestros usuarios no perciban un incidente ocurrido en la base de datos de manera transparente.

Oracle TAF lo podemos implementar sobre versiones Oracle Database 9i y superiores, además que su implementación es sencilla y fácil de integrarse sobre aplicaciones .NET ya puestas en producción.

Existen 2 limitantes que mantiene Oracle TAF y deberían ser evaluadas antes de iniciar su implementación en nuestros proyectos y son:

- TAF recién inicia el failover de sesión al intentar realizar una operación sobre la base de datos utilizando una conexión que ya no está disponible.
- Oracle Database 12c ofrece un nuevo feature llamado Transaction Guard, el cual nos da la posibilidad de obtener información del estado de las transacciones que estuvieron pendiente antes del failover de sesión. Este feature es utilizado en conjunto con FCF y no con TAF.

Publicado por Ing. Francisco Riccio. Es un IT Architect en IBM Perú e instructor de cursos oficiales de certificación Oracle. Está reconocido por Oracle como un Oracle ACE y certificado en productos de Oracle Application & Base de Datos.

e-mail: francisco@friccio.com

web: www.friccio.com