



TUTORIAL APLICACIONES
EVOLUTION CON ACTIVEX
EVOLINK

1	Introducción.....	2
2	Requisitos previos.....	2
3	Desarrollo de una aplicación call center .NET con ActiveX evolink	2
3.1	Crear un proyecto inicial	2
3.2	Añadir la referencia COM Evolink ActiveX Control module	3
3.3	Función para IniciarSesionAgente()	4
3.4	Un botón para iniciar llamadas	5
3.5	Tratamiento de eventos	6
3.6	Resumen	7

1 INTRODUCCIÓN

La interfaz Evolink permite integrar aplicaciones Windows escritas en cualquier lenguaje de programación compatible con ActiveX o .NET, como por ejemplo Microsoft C# o VB.NET, Delphi, etc.

Cuando instalas la aplicación de agente Evolution con setup_iagent.exe, también se instala automáticamente este componente evolink.ocx. Este ActiveX facilita la conexión a un servidor Evolution y acceder a todos los servicios y eventos de la API de agente, con lo que te permitirá desarrollar aplicaciones Windows para call centers.

Este documento ilustra la facilidad que ofrece Evolution para desarrollar aplicaciones en distintos entornos, en este caso en C# .NET. Para ello hemos desarrollado un ejemplo simple facilitando el código de la aplicación.

En el manual de referencia existe más información acerca de las facilidades que ofrece Evolution para el desarrollo de aplicaciones.

2 REQUISITOS PREVIOS

Es necesario tener correctamente instalado Evolution e iAgent, para probar el argumentario.

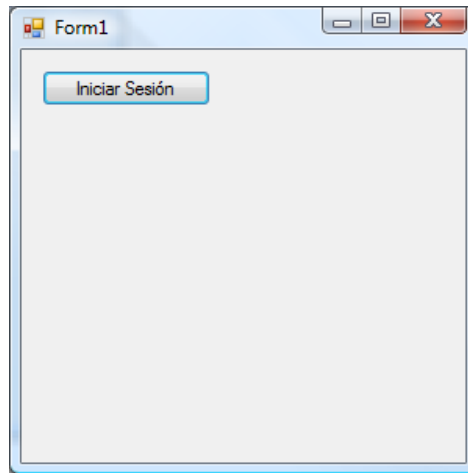
MS Visual Studio 2008 © para revisar el código.

3 DESARROLLO DE UNA APLICACIÓN CALL CENTER .NET CON ACTIVEX EVOLINK

Para que te hagas una idea de la potencia de esta API te propongo desarrollar una aplicación muy simple en C#, siguiendo los siguientes pasos:

3.1 CREAR UN PROYECTO INICIAL

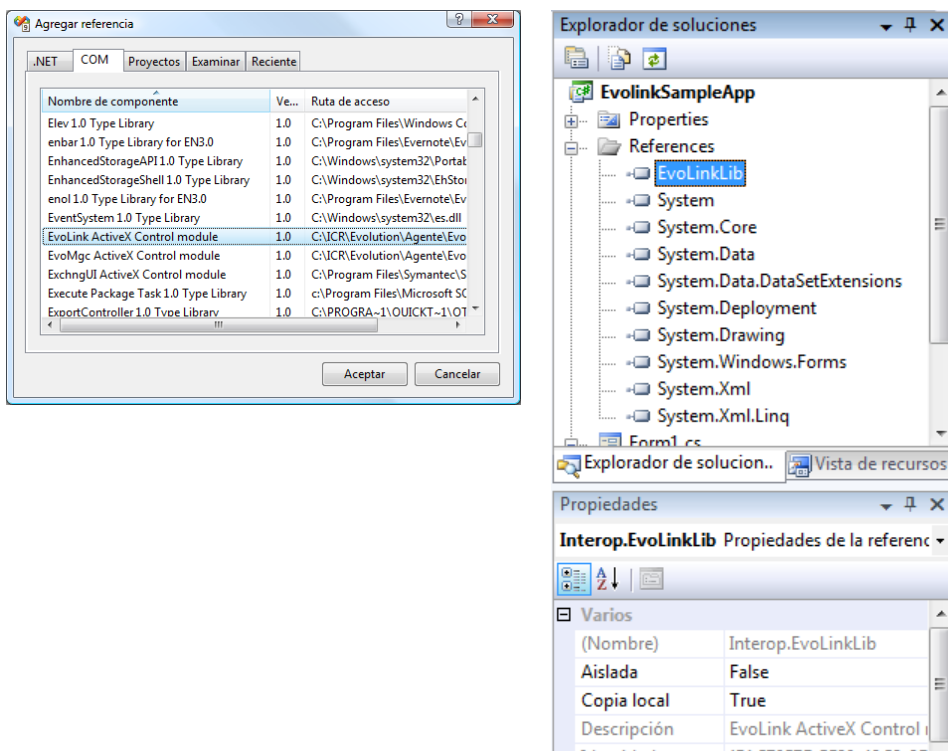
1. Abre Microsoft Visual Studio 2008
2. Crea un nuevo proyecto con "Archivo" → "Nuevo" → "Proyecto". Selecciona un proyecto del tipo "Aplicación de Windows Forms", y pulsa [Aceptar]
3. Si el cuadro de herramientas no está visible, actívalo con el menú "Ver" → "cuadro de herramientas" y arrastra un control del tipo "button" sobre el formulario Form1. Ajustaremos la propiedad "text" para que el botón muestre el texto "Iniciar sesión".
4. Compila y ejecuta el proyecto con F5, para comprobar que la aplicación muestra un diálogo con el botón.



3.2 AÑADIR LA REFERENCIA COM EVOLINK ACTIVEX CONTROL MODULE

Una vez hemos comprobado que el proyecto .NET se compila y ejecuta correctamente, cerramos la aplicación y le añadiremos una referencia a nuestro ActiveX “Evolinkag”:

5. En el explorador de soluciones, pulsa el botón derecho en el nodo “References” y “Agregar referencia”.
6. Aparecerá el diálogo “Agregar referencia”. Selecciona la pestaña “COM” y en la lista de componentes busca “EvoLink ActiveX Control module” y agrégalo pulsando [Aceptar].
7. Bajo el explorador de soluciones aparecerá una nueva referencia “EvoLinkLib”



3.3 FUNCIÓN PARA INICIARSESIONAGENTE()

8. Para acceder al código, pulsa sobre el formulario Form1 con el botón derecho, “Ver código”
9. Añadiremos una directiva `using EvoLinkLib;`
10. En la clase Form1 definiremos un atributo del tipo `EvoLinkAgClass`
11. Inicializaremos este atributo en el constructor del formulario con `m_evotlink = new EvoLinkAgClass();`
12. En el constructor también inicializaremos las propiedades correspondientes a la dirección y puertos IP del servidor Evolution, como por ejemplo:

```
m_evotlink = new EvoLinkAgClass();
m_evotlink.ServerIpAddress = "192.168.0.35";
m_evotlink.ServerTcpPort = 3555;
```

13. También completaremos el formulario con un control tipo “label” para poder mostrar algunos textos al usuario.
14. En el manejador `button1_Click()` añadiremos la funcionalidad de `IniciarSesionAgente()`. Adecúa los valores de los parámetros usuario, contraseña y puesto de trabajo a tu entorno.

```
EvoLinkLib.EInicioSesionAgente rc =
m_evotlink.IniciarSesionAgente("AGENTE", "AGENTE",
                               "my application", "PT418", false);
label1.Text = "IniciarSesionAgente rc=" + rc;
```

El código de ejemplo queda como sigue a continuación:

```
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;

// namespace de evotlink
using EvoLinkLib;
```

```

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        EvoLinkAgClass m_evotlink ;

        int m_currentIdCall;

        public Form1()
        {
            InitializeComponent();

            // inicializamos las propiedades de evolink
            m_evotlink = new EvoLinkAgClass();
            m_evotlink.ServerIpAddress = "192.168.0.35";
            m_evotlink.ServerTcpPort = 3555;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            EvoLinkLib.EInicioSesionAgente rc ;

            rc = m_evotlink.IniciarSesionAgente("AGENTE", "AGENTE",
                                                "my application", "PT418", false);

            label1.Text = "IniciarSesionAgente rc=" + rc;
        }
    }
}

```

3.4 UN BOTÓN PARA INICIAR LLAMADAS

Enriqueceremos un poco nuestro ejemplo añadiendo una caja de texto tipo “textBox” para que el usuario pueda escribir el número de teléfono de destino y un botón para llamar:

15. Añadimos un control de tipo "textBox" al formulario, y también un botón Button2, con un texto "Llamar..."
16. En el manejador button2_Click() añadimos el siguiente código:

```
private void button2_Click(object sender, EventArgs e)
{
    m_evotlink.RealizarLlamada(this.textBox1.Text,
                              ref m_currentIdCall);
    label1.Text = "Llamando...";
}
```

Si recompilamos nuestro proyecto y lo ejecutamos comprobaremos que al pulsar el botón [Llamar...] Evolution inicia una llamada al número de teléfono que indiquemos.

3.5 TRATAMIENTO DE EVENTOS

Pero para poder desarrollar aplicaciones de call center es imprescindible poder manejar eventos telefónicos y de aplicación.

Para gestionarlos usaremos los manejadores de eventos. En nuestro ejemplo trataremos el evento AlertandoLlamadaPrivada, pero de manera parecida podemos tratar el resto de eventos proporcionados por la API

17. Creamos un método AlertandoLlamadaPrivada con unos parámetros compatibles con el delegado del evento AlertandoLlamadaPrivadaEventHandler() que lanza evolink:

```
public delegate void
IEvoLinkAgEvents_AlertandoLlamadaPrivadaEventHandler(int
    IdAgenteLocal, int IdCall, System.DateTime tContacto,
    EvoLinkLib.ESentidoCont TipoContacto, ref string
    DevInterloc, int IdContacto, ref string Dnis)
```

18. Nuestro método AlertandoLlamadaPrivada() se limitará a escribir el número de nuestro interlocutor en la etiqueta.

```
//EvoLinkLib.IEvoLinkAgEvents_AlertandoLlamadaPrivadaEventHandler

public void AlertandoLlamadaPrivada(int IdAgenteLocal,
    int IdCall,
    System.DateTime tContacto,
    EvoLinkLib.ESentidoCont TipoContacto,
    ref string DevInterloc,
    int IdContacto,
    ref string Dnis)

{

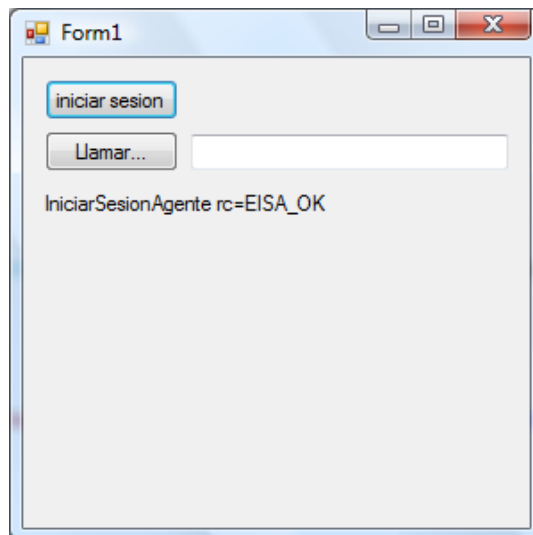
    label1.Text = "AlertandoLlamadaPrivada from " + DevInterloc;

}
```

19. Para que este método se ejecute cuando evolink nos envíe el evento correspondiente deberemos vincularlo al *event handler* correspondiente. Hacer esto es muy sencillo, añadiendo el siguiente código al constructor de Form1:

```
m_evolink.AlertandoLlamadaPrivada += new  
IEvoLinkAgEvents_AlertandoLlamadaPrivadaEventHandler (  
this.AlertandoLlamadaPrivada );
```

Si recompilamos el proyecto y ejecutamos nuestra aplicación podremos comprobar que cuando realizamos o recibimos una llamada “privada” nuestro manejador de evento se ejecuta. Recuerda que las llamadas privadas son aquellas que no son llamadas de campaña ni entre extensiones de agentes.



3.6 RESUMEN

A través de este ejemplo muy sencillo hemos mostrado cómo se puede desarrollar una aplicación .NET que accede a las funciones del servidor y es capaz de detectar y tratar eventos de llamada.

A través del componente Evolinkag.ocx podrás desarrollar aplicaciones de escritorio Evolution en cualquier lenguaje de programación compatible con ActiveX o .NET.

Proponemos que completes el ejemplo con otros manejadores de eventos, así como utilizando otras funciones disponibles en el ActiveX evolink.

Podrás encontrar más documentación sobre la API en el “Manual de Referencia de Evolution”.