



MÓDULO 4

API'S DE INTEGRACIÓN EVOLUTION

1	Introducción	3
2	La interfaz de scripting de iAgent	3
2.1	Listado de funciones	3
2.2	Ejemplo de integración de una aplicación Web	6
2.3	Ejemplo de integración con access 2007	7
3	Integración de aplicaciones con XML	16
3.1	Ejemplo de integración con XML	16
4	Integración de aplicaciones con EvoLinkAg.....	20
4.1	Ejemplo de integración de una aplicación VB.NET	20

1 INTRODUCCIÓN

En este módulo veremos el uso de las diferentes API's de integración de Evolution. Estas API's nos permiten:

- Integrar funcionalidad de Evolution en los argumentarios.
- Integrar aplicaciones de terceros (CRM's, ERP's o nuestra propia aplicación de negocio) con aplicaciones Evolution.
- Incorporar funcionalidades de Evolution a aplicaciones de negocio existentes.

Utilizar una u otra, o varias a la vez, dependerá de la estrategia de integración que queramos seguir y de las capacidades técnicas del equipo de desarrollo.

2 LA INTERFAZ DE SCRIPTING DE IAGENT

iAgent expone una interfaz COM, la cual puede ser accedida desde scripting de cliente. El único requisito es disponer del servidor COM (la aplicación iAgent) y llamarla desde un entorno que permita dialogar con servidores COM. Típicamente, la interfaz de scripting se suele utilizar dentro de las aplicaciones o argumentarios Evolution mediante Visual Basic Script o Javascript. También es bastante habitual utilizar esta interfaz desde aplicaciones que soportan VBA como, por ejemplo, aplicaciones de MS Office.

Nota: la interfaz de scripting queda registrada con el setup de iAgent. En el caso de que no estuviera registrada, se puede forzar el registro invocando a iAgent con el argumento register desde la línea de comandos.

2.1 LISTADO DE FUNCIONES

Listado de métodos expuestos por la interfaz de scripting de iAgent

Función	Retorno	Parámetros
RealizarLlamada	VARIANT_BOOL	LPCTSTR sNumeroTelefono, VARIANT* IdNewCall
RealizarLlamadaEnCampanya	VARIANT_BOOL	LONG idTransaccion, VARIANT* IdNewCall
RealizarLlamadaEnCampanyaEx	VARIANT_BOOL	LONG idTransaccion, LPCTSTR sTelefono, VARIANT* IdNewCall
ColgarLlamada	VARIANT_BOOL	LONG IdCall

TransferirLlamada	VARIANT_BOOL	LONG IdActiveCall, LONG IdHeldCall, VARIANT* IdNewCall
ConferenciarLlamada	VARIANT_BOOL	LONG IdActiveCall, LONG IdHeldCall, VARIANT* IdNewCall
SetInfoTransferencia	VARIANT_BOOL	LONG IdCall, LPCTSTR Info
AsignarValorClave	VARIANT_BOOL	LONG IdTransaccion, LPCTSTR Clave, LPCTSTR Valor
ObtenerValorClave	VARIANT_BOOL	LONG IdTransaccion, LPCTSTR Clave, VARIANT* Valor
IdentificadoSujetoInterloc	VARIANT_BOOL	LONG IdTransaccion, LONG IdSujeto
GetLastCodigoCausa	LONG	
GetLastTextoCausa	BSTR	
EnviarSMS	VARIANT_BOOL	LONG IdTransaccion, LPCTSTR NumeroTelefono, LPCTSTR Mensaje
MostrarLocalizadores		VARIANT_BOOL bShow
MostrarHistoricos		VARIANT_BOOL bShow
AparcarLlamada	VARIANT_BOOL	LONG IdCall
RecuperarLlamada	VARIANT_BOOL	LONG IdCall
MinimizarAplicacion		
MaximizarAplicacion		
RestaurarAplicacion		
PantallaCompletaAplicacion		VARIANT_BOOL bMostrarCompleta
FinalizarTransaccion	VARIANT_BOOL	LONG IdTransaccion, LONG IdFinal, DATE FechaReplanificacion, LONG

		Intervalo, LONG Cuota
FinalizarTransaccion	VARIANT_BOOL	LONG IdTransaccion, LONG IdFinal, DATE FechaReplanificacion, LONG Intervalo, LONG Cuota
ObtenerLlamadasActuales	VARIANT_BOOL	VARIANT* TotalLlamadas
ObtenerIdLlamada	VARIANT_BOOL	LONG Index, VARIANT* IdCall
ObtenerIdTransaccion	VARIANT_BOOL	VARIANT* IdTransaccion
ObtenerEstadoLlamada	VARIANT_BOOL	LONG Index, VARIANT* Estado
ObtenerLlamadaActiva	VARIANT_BOOL	VARIANT* IdCall
AltaSujeto	VARIANT_BOOL	LPCTSTR Nombre, LPCTSTR Apellido1, LPCTSTR Apellido2, LONG CanalOrigen, LPCTSTR Direccion, LPCTSTR Poblacion, LPCTSTRCodigoPostal, LPCTSTR Provincia, LPCTSTR Pais, LONG Idioma, LPCTSTR Telefono, LPCTSTR Fax, LONG LlamarDesde, LONG LlamarHasta, LPCTSTR Email, LPCTSTR Email2, LPCTSTR Movil, LPCTSTR Movil2, LPCTSTR Segmento, LONG IdCampanya, LPCTSTR Dni, LONG CanalPreferencial, LPCTSTR Observaciones, VARIANT_BOOL Contactar, VARIANT* IdSujeto
FinalGestion	VARIANT_BOOL	LONG IdFinal, DATE FechaReplanificacion, LONG Intervalo, LONG Cuota
IniciarPresencial	VARIANT_BOOL	IdCampanya, LONG IdSujeto
IniciarGrabacion	VARIANT_BOOL	
FinalizarGrabacion	VARIANT_BOOL	
MarcarGrabacion	VARIANT_BOOL	LONG nIndiceMarca, LPCTSTR sTextoMarca
ObtenerAgente	OBJETO (1)	

TransferirLlamadaActiva	VARIANT_BOOL	LPCTSTR Telefono
LogTrace		LONG nType, LPCTSTR bstrTrace
ObtenerTransaccion	OBJETO (2)	
ModoCompactoAplicacion		VARIANT_BOOL bMostrarCompacto
MostrarArgumentario		VARIANT_BOOL bMostrar
IniciarSesion	VARIANT_BOOL	Usuario, LPCTSTR Contrasena
FinalizarSesion	VARIANT_BOOL	
MostrarCampanyas		VARIANT_BOOL bShow

2.2 EJEMPLO DE INTEGRACIÓN DE UNA APLICACIÓN WEB

A continuación vemos el uso de dos funciones de la interfaz implementadas como código scripting de cliente en una página HTML. La función Llamar hace uso del método RealizarLlamada para realizar una llamada a la extensión 410. La función FinalGestión finaliza la transacción actual con un final 1.

```
<html>
<script language="vbscript">
  Function Llamar()
    Dim obj,bret,idcall
    Set obj = CreateObject("iAgent.AgentScript")
    bret = obj.RealizarLlamada("410",idcall)
    if not bret then
      alert("Call Error")
    else
      alert("Call Originated")
    end if
    Set obj = nothing
  End Function

  Function FinalGestion()
    Dim obj,bret,idcall, tproxcontacto
    Set obj = CreateObject("iAgent.AgentScript")
    tproxcontacto = now()
    bret = obj.FinalGestion(1, tproxcontacto, 0, 0)
    if not bret then
      alert("Error en FinalGestion()")
    else
      alert("FinalGestion() ok")
    end if
    Set obj = nothing
  End Function
</script>

<body>
<p onclick="Llamar()">pulsa aquí para RealizarLlamada() al 410</p>
<p onclick="FinalGestion()">pulsa aquí para FinalizarGestion() final=1</p>
</body>
</html>
```

2.3 EJEMPLO DE INTEGRACIÓN CON ACCESS 2007

A continuación veremos un ejemplo de argumentario realizado casi en su totalidad con Access 2007. Únicamente es necesario invocar a una página HTML que se encarga de poner en marcha la aplicación desarrollada en Access. Esta aplicación usa VBA para interactuar con Evolution a través de la interfaz de scripting de iAgent.

A nivel de agente, lógicamente, se requiere tener instalado iAgent y Access 2007.

La aplicación Access consta de dos formularios muy sencillos que, básicamente, muestran los datos del cliente y permiten ejecutar un final o reprogramar una llamada. Salvo la página web que pone en marcha Access, el resto de la aplicación está desarrollada íntegramente en un entorno Access 2007.

En primer lugar creamos el código de la página HTML (PopupAccess.htm):

```
<HTML>

<BODY>

<script language="javascript">

    /// variables globales

    var iagent = null;

    /// hide app area...

    ajustarGUI();

    // reopen form if closed

    openAccess() ;

// functions -----
/* inicializar */
function inicializar() {
    if (iagent!=null)
        return true;

    try {
        iagent = new ActiveXObject("iagent.agentscript");
        return true;
    }
}
```

```

catch (e) {
    alert("[inicializar] " + e);
    iagent = null;
    return false;
}
}

function  openAccess ()
{
    var strRuta ="C://icr//evolution//agente//popupaccess2007.accdb";
    var sAplicacionTitle = "popupaccess2007";
    var WshShell = new ActiveXObject("WScript.Shell");

    if (!WshShell.AppActivate(sAplicacionTitle))
    {
        WshShell.run (strRuta, 1);
    }

    var objAccess ;

    try {
        objAccess = GetObject("", "Access.Application");
    }
    catch(e) {
        objAccess = new ActiveXObject("Access.Application");
    }

    strRuta = "C://icr//evolution//agente//popupaccess2007.accdb"
    try { // OpenForm  is notr available now...
        objAccess.DoCmd.OpenForm ("CLIENTES", 0 );
    }
    catch(e) {}
}

```

```

        return false;
    }

function  openAccessManualmente ()
{
    ajustarGUI();
    openAccess();
    return false;
}

function  ajustarGUI ()
{
    /// hide app area...

    /// init iagent (iagent.script) interface
    inicializar();
    iagent.ModoCompactoAplicacion (true);

    iagent.MostrarArgumentario(false);
}
</script>
</BODY></HTML>

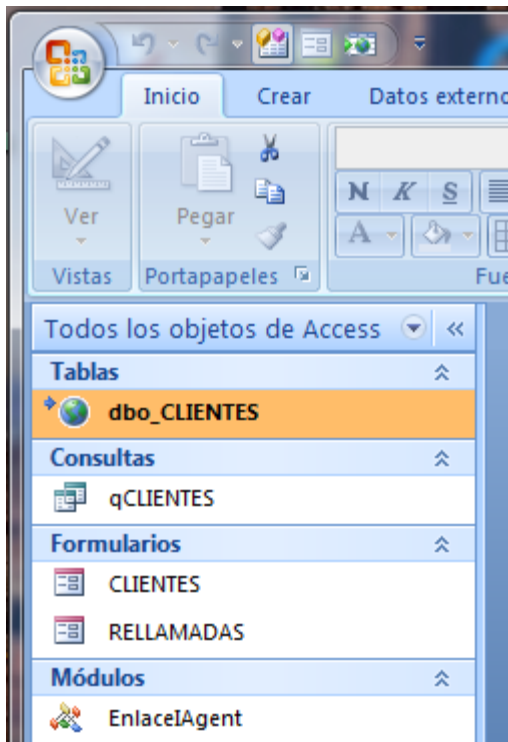
```

La página HTML realiza lo siguiente:

- Habilita la interfaz de usuario de iAgent para ponerla en modo compacto. Esto permite disponer de la botonera funcional de iAgent y deja el resto del escritorio disponible para la aplicación Access 2007.
- A continuación se abre la BD Access C:\icr\Evolution\Agente\popupaccess2007.accdb (si no lo estaba previamente) y se llama al formulario Clientes.

Lo que debemos hacer es ubicar la página web en un servidor web (por ejemplo, en un subdirectorio del directorio args) y dar de alta el argumentario en Manager. Para detalles sobre cómo realizar este paso, consulte el Manual de Referencia de Evolution.

En la base de datos Access es donde se ha de programar la lógica de aplicación. En este ejemplo, la base de datos contiene lo siguiente:



- Una tabla dbo_CLIENES que es un vínculo a la tabla CLIENTES de la BD Evolution
- Una consulta qClientes
- Dos formularios, uno para visualizar los datos del cliente y otro para reprogramar la rellamada.
- Código VBA que implementa la comunicación con la interfaz de scripting de iAgent.

El módulo VBA EnlaceIAgent contiene lo siguiente:

```
Option Compare Database

Public obj As Object

Public Function CrearEnlaceIAgent() As Object

    If obj Is Nothing Then

        Set obj = CreateObject("iAgent.AgentScript")

    End If

    Set CrearEnlaceIAgent = obj

End Function

Public Function AjustarGUI() As Boolean

    If Not bEnlazadoIAgent Then

        bEnlazadoIAgent = True

    End If

End Function
```

```

        CrearEnlaceIAgent

    End If

    Dim obj As Object

    Set obj = CrearEnlaceIAgent

    AjustarGUI = True
End Function

Public Function ObtenerTransaccion() As Variant 'anterior
ObtenerIdentificadorTransaccion

    Dim obj As Object

    Set obj = CrearEnlaceIAgent

    Dim id_transac As Variant

    If obj.ObtenerIDTransaccion(id_transac) Then

        ObtenerTransaccion = id_transac

    End If

End Function

Public Function obtenerInformacion() As Object

    Dim obj As Object

    Set obj = CrearEnlaceIAgent

    Set obtenerInformacion = obj.ObtenerTransaccion()

End Function

Public Function obtenerOperadora() As Variant

    Dim obj As Object

    Set obj = CrearEnlaceIAgent

```

```

    obtenerOperadora = obj.obtenerAgente()

End Function

Public Function obtenerIdCampanya() As Double

    obtenerIdCampanya = obtenerInformacion.idCampanya

End Function

Public Function obtenerIdSujeto() As Double

    Dim obj As Object

    Set obj = CrearEnlaceIAgent

    obtenerIdSujeto = obtenerInformacion.IDSUJETO

End Function

Public Sub finalizarGestion(FINAL As Long, fechaHoraRepla As Date,
rangoPrograma As Long, cuota As Long)

    DoCmd.Close

    Dim obj As Object

    Set obj = CrearEnlaceIAgent

    If Not obj.finalGestion(FINAL, fechaHoraRepla, rangoPrograma, cuota) Then

        MsgBox "ERROR: " & GetLastCodigoCausa & " - " & GetLastTextoCausa,
vbCritical, "ERROR"

        Exit Sub

    End If

End Sub

```

The screenshot shows a web-based form titled 'CLIENTES'. The form is organized into two columns of input fields. The left column includes fields for 'IDORIGINA', 'NOMBRE' (with a dropdown menu), 'APELLIDO1', 'DIRECCION', 'POBLACION', 'CODIGO_PC', 'PROVINCIA', 'PAIS', 'TELEFONO' (with a dropdown menu), 'EMAIL', 'MOVIL', 'OBSERVACIONES', 'LOCALIZABI', and 'LOCALIZABI'. The right column includes fields for 'sDNI', 'sNombreCompleto', 'sEmpresa', and 'cSexo'. Below the input fields, there are four buttons: 'No contesta', 'Comunica', 'Relamar', and 'No llamar mas'. A small red 'X' icon is located at the bottom right of the form area.

El formulario de clientes muestra los datos del cliente actual asociado a la transacción y permite ejecutar algunos finales a partir de los botones correspondientes.

Option Compare Database

```
Private Sub btnComunica_Click()
    finalizarGestion 5, Date, 0, 0
'    Quit
End Sub

Private Sub btnNoContesta_Click()
    finalizarGestion 0, Date, 0, 0
'    Quit
End Sub

Private Sub btnNoLlamarMas_Click()
    finalizarGestion 101, Date, 0, 0
'    Quit
End Sub

Private Sub btnRelamar_Click()
'    form RELAMADAS finaliza con idfinal = 100
    DoCmd.OpenForm "RELLAMADAS"
'    finalizarGestion 100, Date, 0, 0
```

```

''' Quit

End Sub

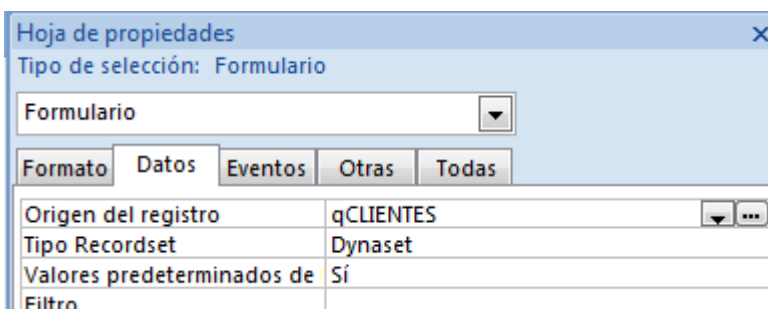
Private Sub Form_Load()

    AjustarGUI

End Sub

```

Nota: ¿Cómo se obtienen los datos del cliente de la transacción actual?



El formulario de cliente obtiene los datos de la consulta qClientes. Esta consulta es la siguiente:

```

SELECT dbo_CLIENTES.*
FROM dbo_CLIENTES
WHERE (((dbo_CLIENTES.IDSUJETO)=obtenerIdSujeto()));

```

La función obtenerIdSujeto se encuentra en el módulo EnalceiAgent y hace uso de la interfaz de scripting de iAgent para saber el identificador del cliente de la transacción actual.

El formulario de rellamadas se llama cuando se selecciona el final de rellamar. Se presenta un diálogo de fecha y hora de próxima llamada al cliente.

```

Option Compare Database

Public vTelAlternativo As Variant

Public bTelAlternativo As Boolean

Private Sub Calendar0_Click()

```

```

Me.txtFecha = Calendar0.Value

End Sub

Private Sub cmdFinProgramada_Click()

    Dim fechaHora As Date

    If IsNull(Me.txtFecha) Or IsNull(Me.txtHora) Then

        MsgBox "SELECCIONA HORA Y FECHA", vbExclamation, "FALTAN DATOS"

        Exit Sub

    End If

    fechaHora = txtFecha & " " & txtHora

    finalizarGestion 100, fechaHora, 60, 0

    '' close parent form!!!

    DoCmd.Close acForm, "CLIENTES"

End Sub

Private Sub Form_Load()

    Me.Calendar0.Value = Date

    bTelAlternativo = False

End Sub

```

3 INTEGRACIÓN DE APLICACIONES CON XML

Otro mecanismo de integración con otras aplicaciones que nos ofrece Evolution es la capacidad de comunicarse con aplicaciones a través de un protocolo XML específico.

Las aplicaciones que desean operar con Evolution a través de este mecanismo, únicamente deben ser capaces de abrir un socket de comunicaciones TCP/IP con el servidor Evolution y, ser capaces de dialogar con este servidor a través del protocolo XML de Evolution.

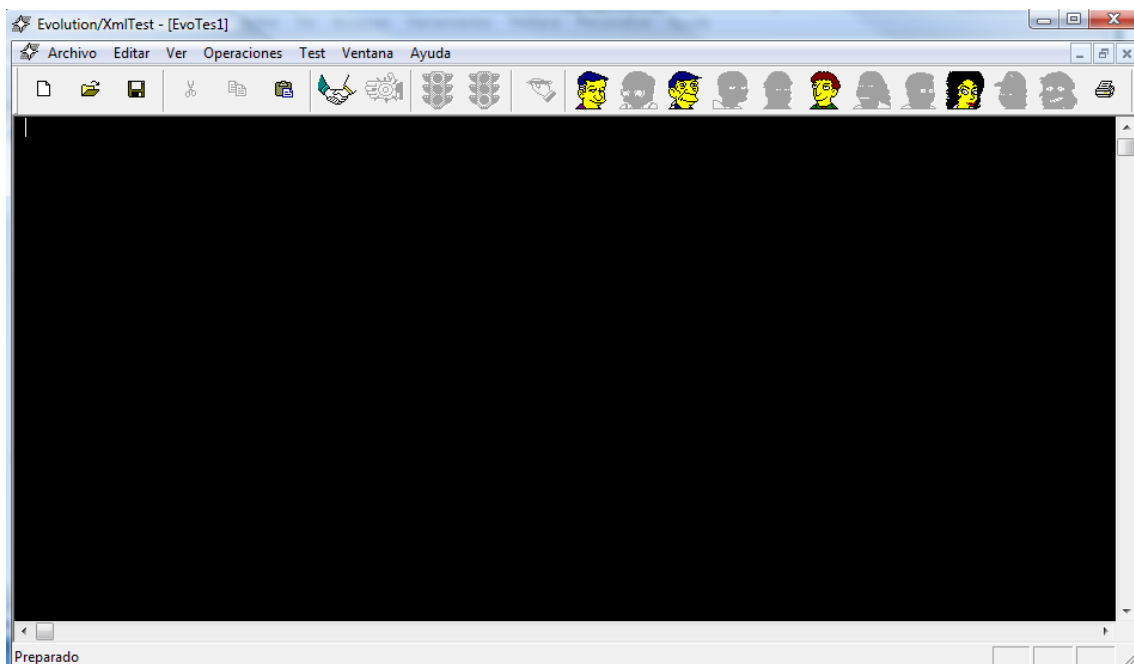
Las especificaciones de este protocolo se encuentran en el SDK de Evolution, disponible para los partners.

Este método proporciona una independencia total de la arquitectura de la aplicación cliente con la arquitectura del entorno Evolution. Es posible integrar aplicaciones que ni siquiera estén desarrolladas en un entorno Windows.

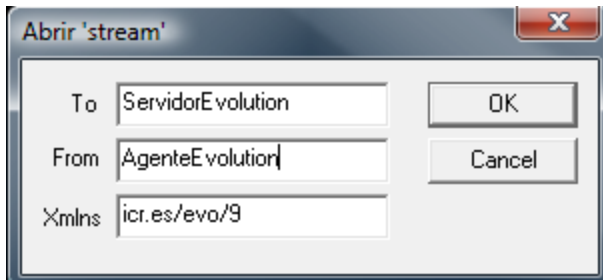
3.1 EJEMPLO DE INTEGRACIÓN CON XML

Utilizaremos la aplicación de uso interno EvoTestXML.exe, que se encuentra en el mismo directorio de instalación de Evoserver. Esta aplicación proporciona una interfaz para generar los mensajes apropiados en formato XML al servidor Evolution y permite asimismo visualizar la respuesta del servidor también en formato XML.

Arrancamos EvoTestXML y nos aparecerá una pantalla como la que sigue:



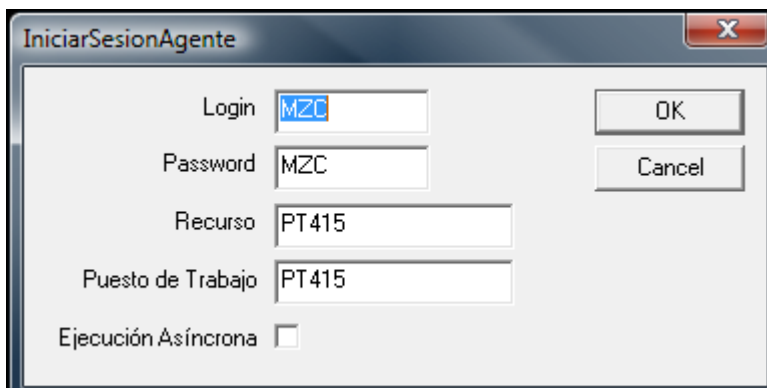
- 1) En el menú Operaciones/Conectar, especificar la IP y puerto del servidor Evolution. Por defecto, el puerto de Evolution es el 3555. Esto únicamente abre un socket TCP/IP con el servidor Evolution.
- 2) Ahora empieza el protocolo XML. Nos sale un diálogo donde nos preguntará To, From y Xmlns. Respondemos:



3) Ahora ya podemos visualizar nuestra petición y la respuesta del servidor en formato XML (se añade a cada línea un "time stamp" para tener una referencia temporal, esto no forma parte del protocolo XML).

```
02/25/10 13:45:54:
<?xml version="1.0"?>
02/25/10 13:45:54:
<stream to="ServidorEvolution" from="AgenteEvolution" xmlns="icr.es/evo/9"/>
02/25/10 13:45:54:
<?xml version="1.0"?>
02/25/10 13:45:54:
<stream to="AgenteEvolution" from="EvoServer@CANPOCH"
xmlns="icr.es/evo/9.4.1411.0"/>
```

4) Ahora vamos a iniciar una sesión de agente. Lo hacemos en el menú Operaciones/Iniciar sesión agente. Debemos introducir un login de agente válido y especificar un puesto de trabajo configurado en Evolution



5) Vemos la petición y la respuesta del servidor

```
02/25/10 13:52:01:
<Request invokeID="2">
  <IniciarSesionAgente>
    <Login>
      MZC
    </Login>
    <Pwd>
      MZC
    </Pwd>
    <Recurso>
      PT415
    </Recurso>
    <PuestoTrabajo>
      PT415
    </PuestoTrabajo>
```

```

        <Asincrono>
          0
        </Asincrono>
      </IniciarSesionAgente>
    </Request>
    02/25/10 13:52:02:
    <Event>
      <NuevoEstadoAgente>
        <Estado>
          144
        </Estado>
        <Texto>
          INACTIVO
        </Texto>
        <CodigoCausa>
          10
        </CodigoCausa>
        <Causa>
          Se ha iniciado la sesión.
        </Causa>
      </NuevoEstadoAgente>
    </Event>
    02/25/10 13:52:02:
    <Response invokeID="2">
      <IniciarSesionAgenteResponse>
        <Resultado>
          1
        </Resultado>
      </IniciarSesionAgenteResponse>
    </Response>

```

5) Ahora podríamos realizar una llamada a otro agente (por ejemplo, llamar a la extensión interna 416). Esto se hace mediante el menú Operaciones/Servicios de telefonía.../Realizar llamada. Vemos la petición y la respuesta y eventos que nos envía el servidor como resultado de dicha petición

```

02/25/10 13:55:06:
<Request invokeID="5">
  <RealizarLlamada>
    <NumTelefono>
      416
    </NumTelefono>
  </RealizarLlamada>
</Request>
02/25/10 13:55:06:
<Response invokeID="5">
  <RealizarLlamadaResponse>
    <Resultado>
      1
    </Resultado>
    <IdNewCall>
      66180
    </IdNewCall>
  </RealizarLlamadaResponse>
</Response>
02/25/10 13:55:09:
<Event>
  <NuevoEstadoAgente>
    <Estado>
      80
    </Estado>

```

```

        <Texto>
            EN_LLAMADA
        </Texto>
        <CodigoCausa>
            120
        </CodigoCausa>
        <Causa>
            Se ha emitido una llamada.
        </Causa>
    </NuevoEstadoAgente>
</Event>
02/25/10 13:55:09:
<Event>
    <AlertandoLlamadaPrivada>
        <IdAgenteLocal>
            10000015
        </IdAgenteLocal>
        <IdCall>
            66180
        </IdCall>
        <tContacto>
            20100225T13:55:08
        </tContacto>
        <TipoContacto>
            2
        </TipoContacto>
        <DevInterloc>
            416
        </DevInterloc>
        <IdContacto>
            100137627
        </IdContacto>
        <Dnis>
            416
        </Dnis>
    </AlertandoLlamadaPrivada>
</Event>

```

6) Finalizamos la sesión mediante Operaciones/Finalizar sesión agente

```

02/25/10 13:57:57:
<Request invokeID="6">
    <FinalizarSesionAgente>
        <null/>
    </FinalizarSesionAgente>
</Request>
02/25/10 13:57:57:
<Response invokeID="6">
    <FinalizarSesionAgenteResponse>
        <Resultado>
            1
        </Resultado>
    </FinalizarSesionAgenteResponse>
</Response>

```

8) Desconectamos la comunicación con el servidor mediante Operaciones/Desconectar. A continuación ya podemos cerrar la aplicación EvoTestXML.

Con este sencillo ejemplo hemos visto que una aplicación externa (EvoTestXML) ha sido capaz de dialogar y realizar alguna operación con el servidor Evolution a través de una simple comunicación TCP/IP y siguiendo un protocolo específico en formato XML.

4 INTEGRACIÓN DE APLICACIONES CON EVOLINKAG

EvoLinkAg es un control ActiveX que forma parte de la suite Evolution y que puede ser utilizado por cualquier aplicación para intercomunicarse con un servidor Evolution. A través de este componente, es posible por tanto integrar aplicaciones capaces de interactuar a través de componentes COM con Evolution. Típicamente, todos los entornos de programación Windows DNA (Visual basic, Visual C++) y Windows .NET (VB.NET, C#, etc.) y otros lenguajes capaces de interactuar a través de ActiveX (Delphi) son capaces de trabajar con este tipo de componentes. Un ejemplo de este tipo de aplicaciones es la propia aplicación de Agente de Evolution, iAgent, que se comunica con el servidor Evolution a través de este componente.

EvolinkAg expone una serie de métodos y funciones que pueden ser utilizados por nuestras aplicaciones para integrarlas con Evolution. La lista completa de todos los métodos proporcionados por EvoLinkAg se encuentra en el Manual de Referencia de Evolution.

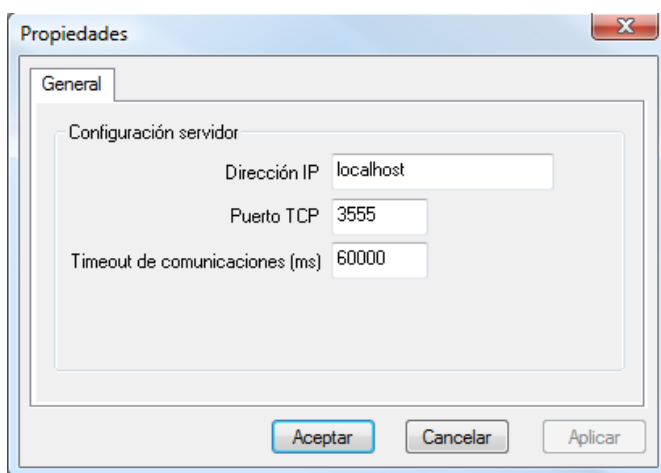
4.1 EJEMPLO DE INTEGRACIÓN DE UNA APLICACIÓN VB.NET

Vamos a crear una aplicación en VB.NET que utiliza EvoLinkAg para integrarse con Evolution.

1) En primer lugar creamos el entorno adecuado para usar EvoLinkAg. Probablemente nos interese agregar este componente COM al cuadro de herramientas de Visual Studio y así poder incorporar este elemento a nuestro proyecto fácilmente.

- Comprobar que el componente EvoLinkAg.ocx está registrado en la máquina de desarrollo.
- Añadir en el cuadro de herramientas el componente COM "EvoLink Agent Control"

2) Una vez hayamos incorporado este componente, podemos arrastrarlo a nuestro formulario para hacer uso de él. El control tiene unas propiedades que podemos editar en tiempo de diseño (y también en tiempo de ejecución) para decirle con qué servidor Evolution queremos conectarnos.



3) Aquí vemos el formulario completo:

Y el código asociado a este Form:

```

Public Class Form1

    Private Sub mostrar_informacion(ByVal msg)
        lMensajes.Items.Add(Date.Today & " - " & Now.Hour & ":" & Now.Minute & " - " &
msg)
        lMensajes.SelectedIndex = lMensajes.Items.Count - 1
    End Sub

    Public Function ExisteNumEnCombo(ByVal Lista As ComboBox, ByVal texto As Long,
ByVal Index As Long)
        ' buscar un numero en una lista y devuelve el indice (index)
        Dim i
        ExisteNumEnCombo = False
        Index = -1
        For i = 0 To Lista.Items.Count - 1
            If Lista.Items(i) = texto Then
                Index = i
                ExisteNumEnCombo = True
            End If
        Next i
    End Function

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        EvoLinkAgl.HabilitarEventos(False)

    End Sub

    Private Sub bLogin_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bLogin.Click
        If bLogin.Text = "Logout" Then
            EvoLinkAgl.FinalizarSesionAgente()
            bLogin.Text = "Login"
            mostrar_informacion("Finalizada Session")
        Else
            If EvoLinkAgl.IniciarSesionAgente(tLogin.Text, tPassword.Text, "recl",
tPuesto.Text, False) Then
                bLogin.Text = "Logout"
                mostrar_informacion("Iniciada Session")
            Else
                mostrar_informacion("No se ha podido iniciar la sesion con el " &
tLogin.Text & " Causa : " & EvoLinkAgl.GetLastCodigoCausa() & " " &
EvoLinkAgl.GetLastTextoCausa())
            End If
        End If
    End Sub

```

```

        End If
    End Sub

    Private Sub bLLamar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bLLamar.Click
        Dim idNewCall As Integer
        Dim Index As Long

        If EvoLinkAgl.RealizarLlamada(tTelefono.Text, idNewCall) Then
            mostrar_informacion("Realizando llamada al telefono " & tTelefono.Text)
            If ExisteNumEnCombo(lTelefonos, idNewCall, Index) Then
                lTelefonos.SelectedIndex = Index
            Else
                lTelefonos.Items.Add(idNewCall)
                lTelefonos.SelectedIndex = lTelefonos.Items.Count - 1
            End If

        Else
            mostrar_informacion("error al realizar la llamada al telefono " &
tTelefono.Text & " Causa : " & EvoLinkAgl.GetLastCodigoCausa() & " " &
EvoLinkAgl.GetLastTextoCausa())
        End If

    End Sub

    Private Sub bContestar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bContestar.Click
        If EvoLinkAgl.AceptarLlamada(lTelefonos.Items(lTelefonos.SelectedIndex)) Then
            mostrar_informacion("llamada " & lTelefonos.Items(lTelefonos.SelectedIndex)
& " contestada ")
        Else
            mostrar_informacion("No se ha podido contestar la llamada " &
lTelefonos.Items(lTelefonos.SelectedIndex) & " Causa : " &
EvoLinkAgl.GetLastCodigoCausa() & " " & EvoLinkAgl.GetLastTextoCausa())
        End If
    End Sub

    Private Sub bColgar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bColgar.Click
        If lTelefonos.SelectedIndex = -1 Then
            Exit Sub
        End If
        If EvoLinkAgl.ColgarLlamada(lTelefonos.Items(lTelefonos.SelectedIndex)) Then
            mostrar_informacion("llamada " & lTelefonos.Items(lTelefonos.SelectedIndex)
& " colgada ")
        Else
            mostrar_informacion("No se ha podido colgar la llamada " &
lTelefonos.Items(lTelefonos.SelectedIndex) & " Causa : " &
EvoLinkAgl.GetLastCodigoCausa() & " " & EvoLinkAgl.GetLastTextoCausa())
        End If
    End Sub

    Private Sub bPausar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bPausar.Click
        If EvoLinkAgl.AparcarLlamada(lTelefonos.Items(lTelefonos.SelectedIndex)) Then
            mostrar_informacion("llamada " & lTelefonos.Items(lTelefonos.SelectedIndex)
& " aparcada ")
        Else
            mostrar_informacion("No se ha podido aparcar la llamada " &
lTelefonos.Items(lTelefonos.SelectedIndex) & " Causa : " &
EvoLinkAgl.GetLastCodigoCausa() & " " & EvoLinkAgl.GetLastTextoCausa())
        End If
    End Sub

    Private Sub bRecuperar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bRecuperar.Click
        If EvoLinkAgl.RecuperarLlamada(lTelefonos.Items(lTelefonos.SelectedIndex)) Then
            mostrar_informacion("llamada " & lTelefonos.Items(lTelefonos.SelectedIndex)
& " recuperada ")
        Else
            mostrar_informacion("No se ha podido recuperar la llamada " &
lTelefonos.Items(lTelefonos.SelectedIndex) & " Causa : " &
EvoLinkAgl.GetLastCodigoCausa() & " " & EvoLinkAgl.GetLastTextoCausa())
        End If
    End Sub

```

```

Private Sub bConferenciar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bConferenciar.Click
    Dim idCallActiva As Long
    Dim IdNewCall As Long
    If EvoLinkAgl.GetLlamadaActiva(idCallActiva) Then
        mostrar_informacion("la llamada a conferenciar sera la llamada activa con
id " & idCallActiva)
        If EvoLinkAgl.ConferenciarLlamada(idCallActiva,
lTelefonos.Items(lTelefonos.SelectedIndex), IdNewCall) Then
            mostrar_informacion("llamada activa " & idCallActiva & " en
conferenciad con el " & lTelefonos.Items(lTelefonos.SelectedIndex) & " con IDNewCall "
& IdNewCall)
        Else
            mostrar_informacion("No se ha podido poner en conferencia la llamada
activa " & idCallActiva & " a " & lTelefonos.Items(lTelefonos.SelectedIndex) & " Causa
: " & EvoLinkAgl.GetLastCodigoCausa() & " " & EvoLinkAgl.GetLastTextoCausa())
        End If
    Else
        mostrar_informacion("No se puede recuperar el ID de la llamada activa" & "
Causa : " & EvoLinkAgl.GetLastCodigoCausa() & " " & EvoLinkAgl.GetLastTextoCausa())
    End If
End Sub

Private Sub bTransferir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bTransferir.Click
    Dim idCallActiva As Long
    Dim IdNewCall As Long
    If EvoLinkAgl.GetLlamadaActiva(idCallActiva) Then
        mostrar_informacion("la llamada a transferir sera la llamada activa con id
" & idCallActiva)
        If EvoLinkAgl.TransferirLlamada(idCallActiva,
lTelefonos.Items(lTelefonos.SelectedIndex), IdNewCall) Then
            mostrar_informacion("llamada activa " & idCallActiva & " Transferida al
" & lTelefonos.Items(lTelefonos.SelectedIndex) & " con IDNewCall " & IdNewCall)
        Else
            mostrar_informacion("No se ha podido transferir la llamada activa " &
idCallActiva & " a " & lTelefonos.Items(lTelefonos.SelectedIndex) & " Causa : " &
EvoLinkAgl.GetLastCodigoCausa() & " " & EvoLinkAgl.GetLastTextoCausa())
        End If
    Else
        mostrar_informacion("No se puede recuperar el ID de la llamada activa" & "
Causa : " & EvoLinkAgl.GetLastCodigoCausa() & " " & EvoLinkAgl.GetLastTextoCausa())
    End If
End Sub

Private Sub lMensajes_DoubleClick(ByVal sender As Object, ByVal e As
System.EventArgs) Handles lMensajes.DoubleClick
    MsgBox(lMensajes.Items(lMensajes.SelectedIndex).ToString)
End Sub

Private Sub EvoLinkAgl_AlertandoLlamadaPrivada(ByVal sender As Object, ByVal e As
AxEvoLinkLib.IEvoLinkAgEvents_AlertandoLlamadaPrivadaEvent) Handles
EvoLinkAgl.AlertandoLlamadaPrivada
    Dim Index As Long

    mostrar_informacion("Alertando llamada privada ID[" & e.idCall & "]" & " en
sentido " & e.tipoContacto & " por telefono " & e.devInterloc & " llamando a " &
e.dnis)
    If ExisteNumEnCombo(lTelefonos, e.idCall, Index) Then
        lTelefonos.SelectedIndex = Index
    Else
        lTelefonos.Items.Add(e.idCall)
        lTelefonos.SelectedIndex = lTelefonos.Items.Count - 1
    End If
End Sub

Private Sub EvoLinkAgl_MensajeAdministrativo(ByVal sender As Object, ByVal e As
AxEvoLinkLib.IEvoLinkAgEvents_MensajeAdministrativoEvent) Handles
EvoLinkAgl.MensajeAdministrativo
    mostrar_informacion("Tenemos un mensaje administrativo tipo : " & e.tipo & " :
" & e.mensaje)
End Sub

Private Sub EvoLinkAgl_SucesoInesperado(ByVal sender As Object, ByVal e As
AxEvoLinkLib.IEvoLinkAgEvents_SucesoInesperadoEvent) Handles
EvoLinkAgl.SucesoInesperado
    mostrar_informacion("Ha ocurrido un el suceso inesperado " & e.codigoCausa & "

```

```

" & e.causa)
End Sub

Private Sub EvoLinkAgl_AlertandoLlamadaEnCampanya(ByVal sender As System.Object,
ByVal e As AxEvoLinkLib.IEvoLinkAgEvents_AlertandoLlamadaEnCampanyaEvent) Handles
EvoLinkAgl.AlertandoLlamadaEnCampanya
Dim Index As Long

mostrar_informacion("Alertando llamada en campanya ID[" & e.idCall & "] del
sujeto" & e.idSujeto & " Para la campanya " & e.idCampanya & " en sentido " &
e.tipoContacto & " por telefono " & e.devInterloc & " llamando a " & e.dnis)
If ExisteNumEnCombo(lTelefonos, e.idCall, Index) Then
lTelefonos.SelectedIndex = Index
Else
lTelefonos.Items.Add(e.idCall)
lTelefonos.SelectedIndex = lTelefonos.Items.Count - 1
End If

End Sub

Private Sub EvoLinkAgl_NuevoEstadoAgente(ByVal sender As Object, ByVal e As
AxEvoLinkLib.IEvoLinkAgEvents_NuevoEstadoAgenteEvent) Handles
EvoLinkAgl.NuevoEstadoAgente
mostrar_informacion("Ha Cambiado el estado del agente a :" & e.estado & " " &
e.texto & " por causa " & e.codigoCausa & " " & e.causa)
End Sub

Private Sub EvoLinkAgl_ColgadaLlamada(ByVal sender As Object, ByVal e As
AxEvoLinkLib.IEvoLinkAgEvents_ColgadaLlamadaEvent) Handles EvoLinkAgl.ColgadaLlamada
mostrar_informacion("se ha colgado la llamada idCall " & e.idCall)
Dim item As Long

If ExisteNumEnCombo(lTelefonos, e.idCall, item) Then
lTelefonos.Items.RemoveAt(item)
mostrar_informacion("se ha eliminado el idCall " & e.idCall)
Else
mostrar_informacion("No se ha eliminado el id de llamada " & e.idCall & "
pues no esta en la lista ")
End If

End Sub

Private Sub EvoLinkAgl_EstablecidaLlamadaAgente(ByVal sender As Object, ByVal e As
AxEvoLinkLib.IEvoLinkAgEvents_EstablecidaLlamadaAgenteEvent) Handles
EvoLinkAgl.EstablecidaLlamadaAgente
mostrar_informacion("Establecida llamada Agente ID[" & e.idCall & "] del
agente" & e.idSujeto & " Para la campanya " & e.idCampanya & " en sentido " &
e.tipoContacto & " por telefono " & e.devInterloc & " llamando a " & e.dnis)
Dim Index As Long
If ExisteNumEnCombo(lTelefonos, e.idCall, Index) Then
lTelefonos.SelectedIndex = Index
Else
lTelefonos.Items.Add(e.idCall)
lTelefonos.SelectedIndex = lTelefonos.Items.Count - 1
End If

End Sub

Private Sub EvoLinkAgl_EstablecidaLlamadaPrivada(ByVal sender As Object, ByVal e As
AxEvoLinkLib.IEvoLinkAgEvents_EstablecidaLlamadaPrivadaEvent) Handles
EvoLinkAgl.EstablecidaLlamadaPrivada
mostrar_informacion("Establecida llamada privada ID[" & e.idCall & "] en
sentido " & e.tipoContacto & "por telefono " & e.devInterloc & " llamando a " & e.dnis)
Dim Index As Long
If ExisteNumEnCombo(lTelefonos, e.idCall, Index) Then
lTelefonos.SelectedIndex = Index
Else
lTelefonos.Items.Add(e.idCall)
lTelefonos.SelectedIndex = lTelefonos.Items.Count - 1
End If

End Sub

Private Sub EvoLinkAgl_EstablecidaLlamadaEnCampanya(ByVal sender As Object, ByVal e
As AxEvoLinkLib.IEvoLinkAgEvents_EstablecidaLlamadaEnCampanyaEvent) Handles
EvoLinkAgl.EstablecidaLlamadaEnCampanya

```

```

    mostrar_informacion("Establecida llamada en campaña ID[" & e.idCall & "] del
    sujeto" & e.idSujeto & " Para la campaña " & e.idCampaña & " en sentido " &
    e.tipoContacto & " por telefono " & e.devInterloc & " llamando a " & e.dnis)
    Dim Index As Long
    If ExisteNumEnCombo(lTelefonos, e.idCall, Index) Then
        lTelefonos.SelectedIndex = Index
    Else
        lTelefonos.Items.Add(e.idCall)
        lTelefonos.SelectedIndex = lTelefonos.Items.Count - 1
    End If
End Sub

Private Sub bConectar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bConectar.Click
    Dim idSesion As Integer
    If bConectar.Text = "Desconectar de Servicio" Then
        EvoLinkAg1.DesconectarDeServicio()
        bConectar.Text = "Conectar a Servicio"
        bDisponible.Text = "No Disponible"
        mostrar_informacion("Desconectado del servicio")
    Else
        bConectar.Text = "Desconectar de Servicio"
        bDisponible.Text = "Disponible"
        'IdServicio hardcoded, modificar con el servicio adecuado
        EvoLinkAg1.ConectarAServicio(100000003, idSesion)
    End If
End Sub

Private Sub bDisponible_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles bDisponible.Click
    If bDisponible.Text = "No Disponible" Then
        EvoLinkAg1.SetEstadoAgente(EvoLinkLib.EEestadoAgente.EEA_DISPONIBLE)
        bDisponible.Text = "Disponible"
        mostrar_informacion("Desconectado del servicio")
    Else
        EvoLinkAg1.SetEstadoAgente(EvoLinkLib.EEestadoAgente.EEA_NO_DISPONIBLE)
        bDisponible.Text = "No Disponible"
    End If
End Sub
End Sub
End Class

```

La funcionalidad prevista en este form es la siguiente:

- Realizar el login/logout de un agente
- Conectarse/desconectarse a un servicio predeterminado (“hardcoded”)
- Realizar una llamada
- Realizar operaciones con una llamada (colgarla, ponerla en espera, recuperarla, conferenciar o realizar una transferencia).
- Poner el agente en modo Disponible/No disponible

Como puede verse, se trata de funcionalidad relacionada con Evolution. Pues bien, todas estas operaciones se realizan haciendo uso de los métodos expuestos por el componente COM EvoLinkAg.

Por ejemplo, para realizar una llamada, el código asociado al evento click del botón de llamar es el siguiente:

```

Private Sub bLLamar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles bLLamar.Click
    Dim idNewCall As Integer
    Dim Index As Long

    If EvoLinkAg1.RealizarLlamada(tTelefono.Text, idNewCall) Then
        mostrar_informacion("Realizando llamada al telefono " & tTelefono.Text)
        If ExisteNumEnCombo(lTelefonos, idNewCall, Index) Then
            lTelefonos.SelectedIndex = Index
        End If
    End If
End Sub

```

```
Else
    lTelefonos.Items.Add(idNewCall)
    lTelefonos.SelectedIndex = lTelefonos.Items.Count - 1
End If

Else
    mostrar_informacion("error al realizar la llamada al telefono " &
tTelefono.Text & " Causa : " & EvoLinkAgl.GetLastCodigoCausa() & " " &
EvoLinkAgl.GetLastTextoCausa())
End If

End Sub
```

Vemos que, en síntesis, realizar la llamada consiste en llamar al método RealizarLlamada de EvoLinkAg.

También podemos ver en el código de este formulario una serie de manejadores de eventos relacionados con la telefonía. Por ejemplo, el evento AlertandoLlamadaPrivada es un evento señalado por el componente EvoLinkAg y que debemos interpretar como que el agente está recibiendo una llamada privada (en este caso, nos limitamos a mostrar el evento en la lista de eventos del formulario).

Extrapolando este ejemplo, podemos incorporar funciones de Evolution a nuestras aplicaciones existentes mediante el uso de EvoLinkAg. Tómese su tiempo para estudiar y probar esta aplicación.