Programmierpraktikum

LVA.Nr. 384.147

WS 03

Implementierung eines Softphones als ActiveX-Control auf Basis des Microsoft Real Time Communications (RTC) Client Application Programming Interfaces (API)

Wolfgang Hatzinger
Matr.Nr. 8926879, StudZw. E754

Inhaltsverzeichnis

Abbildungsverzeichnis

RTC Client API ActiveX Softphone

1. Aufgabenstellung

Es soll mit Hilfe des Microsoft RTC Client Application Programming Interface ein einfaches Softphone als ActiveX-Control implementiert werden. Eingebunden in einer HTML-Seite sollen mit diesem Steuerelement Anrufe über das Internet initiiert und empfangen werden können. Primäres Ziel dieser Aufgabe ist neben der Entwicklung der beschriebenen Anwendung das Kennenlernen und die Handhabung der zur Verfügung stehenden Entwicklungsumgebung. Als Einstiegshilfe in diese Technologie sollen zu Beginn die der Dokumentation beiliegenden Beispiele übersetzt und gestartet werden.

2. Beschreibung des RTC Client API

Das RTC Client Application Programming Interface der Firma Microsoft ist ein Sammlung von COM (Component Object Model) Interfaces und Methoden, mit deren Hilfe PC-PC, PC-Phone und Phone-Phone Audio/Video-Verbindungen oder rein textuelle Instant Messaging (IM)-Sitzungen über das Internet erzeugt werden können. Dieses Softwarepaket ist kostenlos im Internet unter http://download.microsoft.com/download/1/e/5/1e55f794-0a9a-43c4-8698-72e04c97ff72/RtcApiSdk.msi erhältlich.

Es existieren zur Zeit die Versionen 1.0, 1.01 und 1.2, die sich im Funktionsumfang und im unterstützten Betriebsystem unterscheiden. Die Basisversion 1.0 läuft unter Windows XP oder Windows Server 2003 und bietet grundsätzliche SIP-basierte Echtzeitkommunikations-Funktionalität zur Entwicklung von Client-Anwendungen in Visual Basic oder Visual C++. Mit Version 1.01 – auf Windows Server 2003 oder Windows XP SP 1 Plattformen lauffähig – wurde der Umfang um IP-Port und -Adressmapping Funktionalitäten zur Passage von Firewalls und NATs erweitert. Die vorerst letzte Version 1.2 kann auch unter Windows 2000 eingesetzt werden und bietet neben Multiparty Instant Messaging, SIP User Search, Call Forward, Call Transfer, Call Hold und der Verschlüsselung der Mediadaten bei PC-PC Sitzungen noch viele weitere Eigenschaften.

Die Hauptkommunikationsobjekte des RTC Clients sind das in Abbildung 1 gezeigte Client-, das Session- und das Participant-Object. Das RTC Client-Object legt die erlaubten Sitzungsarten und Sitzungsparameter wie die bevorzugten Audio/Video-Geräte, die Lautstärke und die Medientypen fest. Angelegt wird es in Visual Basic durch `Set Objektvariable As New RTCClient`. Das IRTCClient-Interface erzeugt das Session-Object und referenziert auf das IRTCClientPresence- und das IRTCClientProvisioning-Interface.

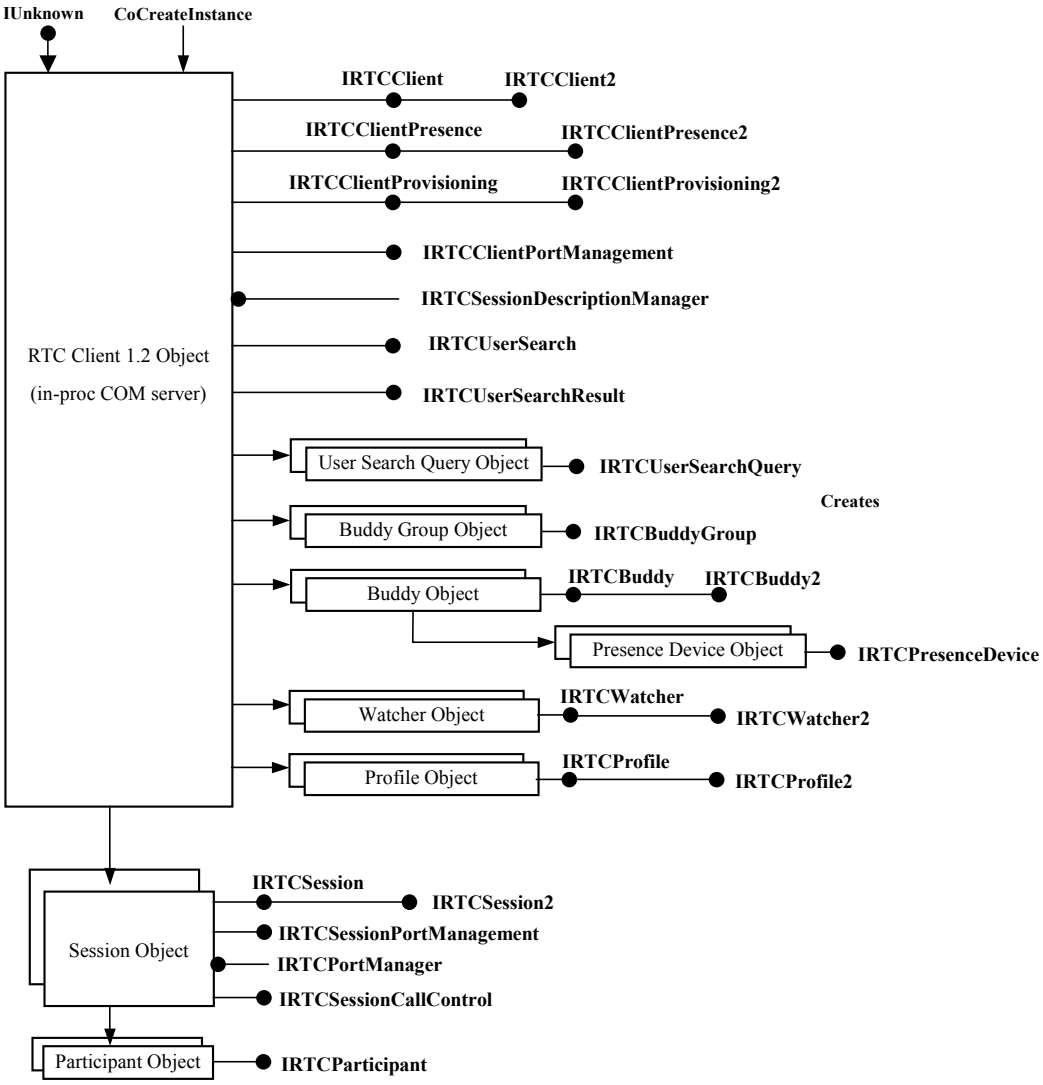

Abbildung 1: RTC Version 1.2 Objekte und Interfaces

Das Session-Object unterstützt alle Sitzungsarten (PC-PC, PC-Phone...) und repräsentiert die Einheit, die Audio/Video-Verbindungen oder IM-Sitzungen erzeugen und empfangen kann. Informationen über die Sitzung wie Medientypen, den augenblicklichen Sitzungszustand oder Teilnehmerinformationen können über das Session-Interface empfangen werden. Mit der Methode AddParticipant wird ein Teilnehmerobjekt (Participant-Object) erzeugt und die Sitzung initiiert. Mehr als zwei Teilnehmer können nur bei „Phone to Phone“ und „Multiparty IM“ Sitzungen teilnehmen. Über das Interface des Participant-Objects können der Teilnehmername, dessen URI und sein Zustand empfangen werden.
Die Objekte zur Verwaltung der Kontaktinformationen sind das Buddy- und das Watcher-Object. Mit dem Provisioning-Interface des Clients werden XML-Profile gesetzt und gelöscht, mit dem Profile-Object-Interface können Informationen aus diesen abgerufen werden.

3. Anleitung zur Installation des RTC Client API unter Windows XP

Zusätzlich zum RTC Client API müssen auch Teile des Microsoft Platform Software Development Kit (PSDK) und zumindest eine der Programmiersprachen Visual C++ oder Visual Basic installiert werden. Dabei spielt die Reihenfolge der Installation eine wesentliche Rolle. Sie ist wie folgt einzuhalten.

(1) Microsoft Visual Studio 6.0

→ Visual Basic 6.0 Enterprise Edition installieren
→ Visual C++ 6.0 Enterprise Edition installieren
→ ActiveX Komponenten installieren

(2) Bei Visual C++ müssen zusätzlich zur Standardinstallation auch die MFC Unicode Bibliotheken und Binaries installiert werden.

Start Button → *Einstellungen* → *Systemsteuerung* → *Software* → *Microsoft Visual Studio* markieren → *Ändern/Entfernen* → *Hinzufügen/Entfernen* → *Visual C++* markieren → *Option ändern* → *MFC und Vorlagebibliotheken für VC++* markieren → *Option ändern* → *MS Foundation Class Bibliotheken* markieren → *Option ändern* → *Statische Bibliotheken* und *Gemeinsam genutzte Bibliotheken für Unicode* markieren → *OK* → *OK* → *OK* → *Weiter*.

(3) Damit die Umgebungsvariablen korrekt konfiguriert sind, muss Visual C++ zumindest einmal vor der Installation des Core SDK gestartet worden sein!

(4) Microsoft Platform Software Development Kit (PSDK) installieren

→ *Core SDK* → *Install this SDK* → *Continue* → *Accept.*

(5) Nach erfolgreicher Installation müssen die PSDK Verzeichnisse noch im Visual Studio registriert werden.

Start Button → *Programme* → *Microsoft Platform SDK* → *Visual Studio* Registration → *Register PSDK Directories with Visual Studio*.

(6) Microsoft Real Time Communicator (RTC) Client v1.2 API installieren

→ Starte `RtcApiSDK.msi`

→ Starte `RtcApiSetup.exe` im neu erzeugten Ordner „`RTC Client API v1.2 SDK`"

4. Übersetzung und Start der RTC Client API Beispiele

Das mitgelieferte Visual Basic Beispiel im Verzeichnis „`RTC Client API v1.2 SDK\SDK\Samples\VBSample\`" lässt sich durch einfaches Doppelklicken des Visual

Basic Project File „`RTCSampleVB.vbp`“ starten. Nach dem Öffnen der Visual Basic Entwicklungsumgebung wird der Source-Code durch anklicken des *Play-Button* gestartet.

Die im Verzeichnis „`RTC Client API v1.2 SDK\SDK\Samples\`“ enthaltenen Visual C++ Beispiele sind schwieriger zu starten.

- Zuerst muss die Plattform SDK Build Environment für Windows XP 32 Bit (Retail) geöffnet werden. *Start Button → Programme → Microsoft Platform SDK February 2003 → Open Build Environment → Windows XP 32-bit Build Environment → Set Windows XP 32-bit Build Environment (Retail).*
- Danach muss ein Pfad auf die Include Files im INC Unterverzeichnis gelegt werden (Bsp.: `PATH C:\Programme\RTC Client API v1.2 SDK\INC;%PATH%`) Die Angabe von %PATH% ist nötig, da sonst die vorhandene Pfadangabe durch den angegebenen Pfad nicht ergänzt, sondern ersetzt wird.
- Im SDK-Build Environment wird in jenes Verzeichnis gewechselt, in dem sich der Source-Code des entsprechenden Beispiels befindet (Bsp.: `cd Samples\netds\rtc\Client\CPP\rtcsample`).
- Durch Eingabe von „`nmake`“ auf der Kommandozeile des SDK Build Environments wird automatisch ein EXE-File gebildet.
  Beachte: Kommt es zu einer Fehlermeldung, liegt das möglicherweise daran, dass die MFC Unicode Binaries, die zum Erzeugen und Ausführen der Beispiele benötigt werden, nicht installiert sind (siehe Installationsanweisungen).
- Im neu angelegten Unterverzeichnis `XP_RETAIL` befindet sich das erzeugte EXE-File, das durch Doppelklicken gestartet werden kann.

5. Erstellen neuer RTC Client v1.2 Applikationen unter Windows XP

Zum Erstellen eigener Beispiele, die das RTC Client API verwenden, muss in Visual Basic 6.0 die entsprechende Library eingebunden werden. Nach dem Start der Visual Basic Entwicklungsumgebung und Auswahl des neu zu erstellenden Projekttyps gelangt man über *Projekt → Verweise* zur Registrierkarte *Verfügbare Verweise*, in der die RTC Core 1.2 Type Library markiert werden muss.
Will man auf einem Windows XP Rechner Programme schreiben, die RTC Version 1.2 verwenden, so muss man mit regsvr32 die `RtcDll.dll` Library registrieren und im gleichen Verzeichnis in dem sich das VB-Projektfile (*.vbp) befindet, ein Manifestfile erstellen. Am einfachsten kopiert man eines der Manifestfiles aus dem `Samples` Ordner und passt es an das eigene Projekt an. Das folgende Listing zeigt ein derartiges Manifestfile.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<assemblyIdentity
version="1.0.0.0"
processorArchitecture="x86"
name="Der Name des Projekts also des ausführbaren Files"
type="win32"
/>
<description>Kurze Beschreibung der Anwendung</description>
<dependency>
<dependentAssembly>
<assemblyIdentity
type="win32"
name=" Microsoft.Windows.Networking.RTCDLL"
version=" 5.2.1.0000"
processorArchitecture="X86"
publicKeyToken="6595b64144ccf1df"
language="*"
/>
</dependentAssembly>
</dependency>
</assembly>
```

Abbildung 2: Beispiel eines Manifestfiles

In diesem File sind mehrere Änderungen vorzunehmen.

1. Im `<assemblyIdentity>` Element sind die Attribute „version“ und „name“ anzupassen.

- Unter „version“ ist die aktuelle Versionsnummer des Projekts einzutragen. Diese ist in Visual Basic unter *Projekt* → *Eigenschaften von Projekt...* → *Erstellen* nachzulesen.
- Bei „name“ ist der Name des VB-Projektfiles ohne Extension einzutragen.

2. Im `<description>` Element folgt eine kurze Beschreibung der Anwendung.

3. Im `<dependentAssembly>` Element ist das „version“-Attribut auf die installierte Rtcdll.dll einzustellen. Die aktuelle Version findet man im Windows-Verzeichnis unter `\winsxs\x86_Microsoft.Windows.Networking.RtcDll_6595b64144ccf1df_5.2.2.1_x-ww_d6bd8b93`. Im hier angeführten Beispiel wäre die Version 5.2.2.1.

4. Gespeichert wird das Manifestfile unter gleichen Namen und im gleichen Ordner wie die Anwendung plus der Extension „.manifest“ (Bsp.: Example.exe → Example.exe.manifest).

6. Implementation eines einfachen SIP-Phones als ActiveX-Control

Ein ActiveX-Control (ActiveX-Steuerelement) ist wie ein von objektorientierten Programmiersprachen bekanntes Objekt aufgebaut. Sein Verhalten wird in Methoden abgebildet, Eigenschaften (Properties) beschreiben den Zustand des Objekts, und die Kommunikation mit dem Objekt wird über Ereignisse (Events) – wie beispielsweise Benutzereingaben – gesteuert.

ActiveX-Steuerelemente enthalten ausführbaren Binärcode, wodurch sie nur in einer abgestimmten Systemumgebung – dem so genannten Steuerelementcontainer – lauffähig sind. Eine ganze Reihe verschiedener Objekte ist am Entstehen eines Controls beteiligt. Die wichtigsten sind:

- Das eigene Control-Objekt
- Das UserControl-Objekt
- Das Ambient-Objekt
- Das Extender-Objekt
- Konstituierende Control-Objekte.

Abbildung 3 zeigt die Objektbeziehungen eines ActiveX-Controls (*MyControl*) mit seinem Container (*MyTestForm*).

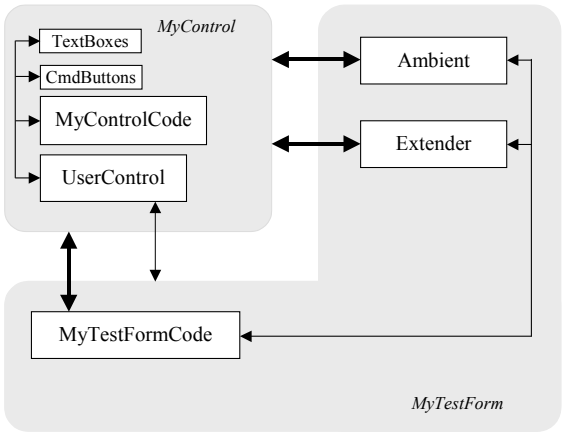

Abbildung 3: Das ActiveX-Control Objekt-Modell

Der einzige Teil des ActiveX-Controls, den man als Autor selbst anlegt, ist das eigene Control-Objekt (*MyControl*). Hier werden neben dem Verhalten des Controls auch alle öffentlichen Methoden, Eigenschaften und Ereignisse definiert, die zur Verwendung des Controls zur Verfügung stehen. Neben den selbstimplementierten Methoden und Eigenschaften hat man auch Zugriff auf die Eigenschaften des UserControl-, des Extender- und des Ambient-Objekts sowie auf jene der konstituierenden Controls, die auf dem eigenen platziert sind. Das Ambient und Extender Objekt ermöglichen dem ActiveX-Steuerelement den Zugriff auf Informationen über den Container, wohingegen das UserControl-Objekt sämtliche Schnittstellen, die das ActiveX-Steuerelement zur Kommunikation mit dem Container benötigt, enthält. Visual Basic und Windows leiten Maus- und Tastatureingaben an das UserControl-Objekt weiter, die dort verarbeitet werden und im selbst erstellten Control-Code beliebig behandelbare Ereignisse auslösen.

Die Erzeugung eigener ActiveX-Controls ist in Visual Basic sehr einfach. Nach dem Öffnen der Entwicklungsumgebung hat man im Fenster „Neues Projekt“ lediglich das Icon zur Erstellung eines ActiveX-Controls auszuwählen. Auf dem so geöffneten Control lassen sich wie bei einem herkömmlichen EXE-Projekt Standardobjekte wie Command-Buttons, Text-Boxes oder Combo-Boxen platzieren und mit Code versehen. Das Standardobjekt eines ActiveX-Steuerelements ist das UserControl-Objekt, das von Visual Basic automatisch angelegt wird, und zahlreiche Ereignisse und Eigenschaften besitzt. Die wichtigsten Ereignisse sind der Initialize- und der Terminate-Event. Die Behandlungsroutine des Initialize-Events ist die erste Prozedur, die nach dem Öffnen des Steuerelements aufgerufen wird. In ihr werden alle globalen Variablen und Objekte des Projekts angelegt und initialisiert. Das Gegenstück dazu bildet die zuletzt ausgeführte *UserObject_Terminate* Prozedur in der alle Objekte wieder freigegeben werden.

Die Benutzeroberfläche des erstellten Softphone-Clients zeigt Abbildung 4. Nach dem Start des Clients lassen sich die Sitzungsart und die gewünschte Zieladresse eingeben und über den Connect-Button eine Verbindung herstellen. Ist die Option Auto Answer markiert, werden eingehende Anrufe auch ohne Drücken des Answer-Buttons entgegen genommen. Mit Hilfe des Wizard-Buttons lassen sich die Audio/Video-Geräte am lokalen Rechner konfigurieren. Über den aktuellen Zustand der Verbindung und des Clients gibt die Statuszeile am unteren Rand des Softphones Auskunft.

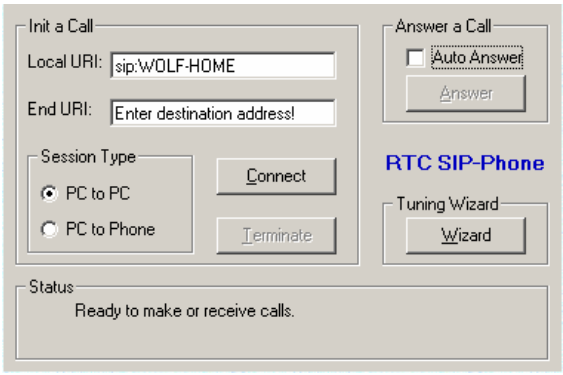

Abbildung 4: Screen Shot des implementierten ActiveX SIP-Phones

Ist eine Verbindung zustande gekommen, verschwindet der Wizard-Button sowie die darüberliegende Beschriftung und ein KeyPad zum Erzeugen von DTMF-Wahltönen wird eingeblendet. Dieses lässt sich auch durch Anklicken des blauen „RTC SIP-Phone“ Schriftzuges einblenden.
Das erstellte Visual Basic Programm befindet sich am Ende des Protokolls und ist durch die zahlreichen eingefügten Kommentare sehr gut lesbar. Das Programm besteht im Wesentlichen aus drei Teilen:

- Dem Deklarations- und Initialisierungsteil

- Den Event-Handlern für die RTC-Client und User-Action Ereignisse
- Und dem Shut-Downteil zur geordneten Freigabe der Objekte.

Das zugrundegelegte Struktogramm zeigt Abbildung 5.

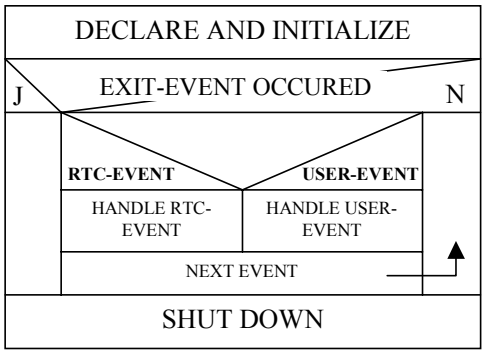

Abbildung 5: Struktogramm des ActiveX SIP-Phones

Deklarations- und Initialisierungsteil

Im Deklarationsteil werden die globalen Konstanten und Objektvariablen für die RTC Client Objekte deklariert. Der Initialisierungsteil wird durch die *UserControl_Initialize*-Prozedur gebildet. In dieser werden das RTC Client Objekt erzeugt und initialisiert, das Ausgangsinterface, das die spezifizierten Event-Typen empfängt, referenziert und die Portzuordnung für ausgehende und eingehende Anrufe festgelegt. Zusätzlich können hier noch die bevorzugten Medientypen gesetzt und die nötigen Einstellungen am User Interface vorgenommen werden.

Event-Handler

Der Handler für die RTC-Client Events wird durch die *g_objRTCClientWithEents_Event* Prozedur gebildet. An diese Subroutine werden die im Initialisierungsteil spezifizierten Client Events gesendet. Die einzige Aufgabe dieser Prozedur besteht darin, die richtige Methode zum aufgetretenen Ereignis aufzurufen. Im vorliegenden Beispiel wird nur auf den `RTCE_SESSION_STATE_CHANGE`-Event getriggert und bei Eintritt dieses Ereignisses die Subroutine *SessionStateEvent* aufgerufen, die ihrerseits entsprechend des übergebenen Sitzungszustands reagiert.
Der Event-Handler für die User-Action-Ereignisse wird durch die einzelnen Click-Methoden der konstituierenden Standard-Objekte (den Command Buttons und Option Boxes) gebildet.

Shut-Down-Teil

Der Shut-Down-Teil wird durch die *UserControl_Terminate*-Prozedur gebildet. In dieser werden noch geöffnete Sitzungen geschlossen, das RTC Client Objekt geregelt heruntergefahren und abgeschalten und zuletzt der allozierte Speicher wieder freigegeben.

Zusätzlich zu den oben genannten Prozeduren gibt es noch die unterstützenden Subroutinen *EnableKeyPad*, *DisableKeyPad* und *ErrorAusgeben*, deren Namen bereits deren genaue Funktion beschreiben.

7. ActiveX-Control und HTML

Der Microsoft Internet Explorer oder Netscapes Navigator ab Version 3.0 bieten Containerdienste zur Ausführung von ActiveX-Steuerelementen an.
Im Microsoft Internet Explorer beispielsweise lassen sich ActiveX-Controls mit Hilfe des HTML-Tags <Object...> in Webseiten einbinden und zusätzlich über Visual Basic Script (VBScript) oder Java Script mit Logik versehen. Greift der Anwender auf eine derartige HTML-Seite zu, wird das Steuerelement, sofern es auf dem lokalen Rechner noch nicht vorhanden ist, vom Web-Server auf den PC heruntergeladen. Trifft ein Web-Browser, der keine Containerdienste unterstützt auf das Object-Tag, ignoriert er es.
Das in Abbildung 6 dargestellte Listing zeigt ein einfaches Beispiel für die Einbindung eines ActiveX-Controls in eine HTML-Seite.

```
<html>
<title>HTML Control Example </title>
<body>
<h1>Mein erstes ActiveX-Control Beispiel</h1>
<p>
<object classid="CLSID:79176FB0-B7F2-11CE-97EF-00AA006D2776"
 codebase="http://microsoft.com/activex/controls/spin32.ocx"
 name=SpinButton1 id=SpinButton1
 width=100 height=121 HSPACE=85>
</object><script language="VBScript">
Sub SpinButton1_SpinUp()
       MsgBox "Up arrow clicked"
End Sub
Sub SpinButton1_SpinDown()
       MsgBox "Down arrow clicked"
End Sub
</script>
</p>
</body>
</html>
```

Abbildung 6: HTML-Einbindung eines ActiveX-Controls

Nach der Überschrift „Mein erstes ActiveX-Control Beispiel“ verwendet dieses HTML-Dokument das HTML-Tag <Object>, um das ActiveX-Steuerelement „Spin Button“ zu laden. Das Tag <Object> legt die Klassenkennung (`classid`), die Steuerelementposition (`hspace`) und Größe (`height, width`) sowie den Pfad fest, von dem das Steuerelement bei Bedarf per Internet herunterzuladen ist. Bevor der Browser ein gefordertes Steuerelement lädt und lokal registriert, überprüft er anhand der per CLSID angegebenen Klassenkennung, ob dieses bereits in der lokalen Registry verwaltet wird. Ist dies nicht der Fall, erfolgt das Herun-

terladen und die lokale Registrierung. Wo das geforderte ActiveX-Steuerelement gefunden werden kann, wird durch das Attribut „codebase“ angegeben. Zusätzlich zeigt das Beispiel die Verwendung von VBScript.

8. Programmlisting

```
Option Explicit

'----- Deklaration der verwendeten Medien Typen -----

Const RTCMT_AUDIO_SEND = &H1
Const RTCMT_AUDIO_RECEIVE = &H2

'----- Verwendete Event Filter -----

Const RTCEF_CLIENT = &H1
Const RTCEF_SESSION_STATE_CHANGE = &H4

'----- Deklaration der globalen RTC Client API Objekte -----

Public g_objRTCClient As RTCClient
Private g_objSession As IRTCSession
Private g_objParticipant As IRTCParticipant
Private WithEvents g_objRTCClientWithEvents As RTCClient

'----- Diese Prozedur wird unmittelbar nach dem Start ausgeführt. Hier werden -----
'----- alle Variablen auf Modulebene initialisiert, die Eigenschaften der     -----
'----- Standardelemente auf ihre Standardwerte gesetzt und das User Interface -----
'----- initialisiert.                                                         -----

Private Sub UserControl_Initialize()

On Error GoTo CheckError

    '----- Anlegen eines neuen RTCClient Objekts -----

    Set g_objRTCClient = New RTCClient

    '----- Initialisieren des RTCClient Objekts -----

    Call g_objRTCClient.Initialize

    '----- Setzen der EventFilter -----

    g_objRTCClient.EventFilter = _
                        RTCEF_CLIENT Or _
                        RTCEF_SESSION_STATE_CHANGE

    ' ----- Referenzieren des die Ereignisse empfangenden Interfaces -----

    Set g_objRTCClientWithEvents = g_objRTCClient

    ' ----- Achte auf eintreffende Anrufe sowohl auf dynamischen als auch -----
    ' ----- auf den statischen well-known Ports                           -----

    g_objRTCClient.ListenForIncomingSessions = RTCLM_BOTH

    ' ----- Festsetzen der verwendeten Mediatypen -----

    Call g_objRTCClient.SetPreferredMediaTypes(RTCMT_AUDIO_SEND Or _
```

```
                                                RTCMT_AUDIO_RECEIVE, True)

    ' ----- Anzeigen der lokalen User-URI und Abfragen der Zieladresse -----

    txtLocalURI.Text = g_objRTCClient.LocalUserURI
    txtEndURI.Text = "Enter destination address!"

    ' ----- Statusausgabe -----

    lblStatus.Caption = "Ready to make or receive calls."

    ' ----- Aktivieren/Deaktivieren der Eingabe-Buttons -----

    cmdTerminate.Enabled = False
    cmdAnswer.Enabled = False

    Exit Sub

CheckError:
    Call ErrorAusgabe
Resume Next

End Sub

'----- Die Terminate Prozedure wird vor dem Schließen des Objekts ausgeführt. -----
'----- Hier werden alle Objekte, die das Steuerelement verwendet, freigegeben.-----

Private Sub UserControl_Terminate()

On Error Resume Next

    ' ----- Terminieren aller offenen Sitzungen -----

    If Not (g_objSession Is Nothing) Then
        g_objSession.Terminate (RTCTR_SHUTDOWN)
    End If

    ' ----- Beenden des RTCClient Objekts -----

    If Not (g_objRTCClient Is Nothing) Then

        '----- Fuer einen geregelten Shutdown des RTC Clients muss zuerst die -----
        '----- "PrepareForShutdown"-Methode des Client-Objekts aufgerufen     -----
        '----- werden und auf den RTCClientEvent "RTCCET_ASYNC_CLEANUP_DONE"- -----
        '----- Event gewartet werden, bevor die IRTCClient.Shutdown-Methode   -----
        '----- zum Beenden aufgerufen wird.                                   -----

        Call g_objRTCClient.PrepareForShutdown
        Call g_objRTCClient.Shutdown

    End If

    ' ----- Nach dem geregelten Shutdown kann der Speicher freigeben werden -----

    Set g_objSession = Nothing
    Set g_objParticipant = Nothing
    Set g_objRTCClientWithEvents = Nothing
    Set g_objRTCClient = Nothing

End Sub

'----- Diese Prozedur wird zum Empfang der Ereignisse benötigt. Sämtliche     -----
'----- Ereignisse werden an diese Prozedur gesendet. Die richtige Methode zum -----
'----- empfangenen Ereignis aufzurufen, ist Aufgabe des Programmierers.       -----
```

```
Private Sub g_objRTCClientWithEvents_Event(ByVal RTCEvent As RTCCORELib.RTC_EVENT,
                                           ByVal pEvent As Object)

    ' ----- Verzweigung zu den entsprechenden Ereignisbehandlungs Prozeduren -----

    Select Case RTCEvent

        Case RTCE_SESSION_STATE_CHANGE
            Call SessionStateEvent(pEvent)

    End Select

End Sub

Private Sub SessionStateEvent(ByVal pEvent As Object)

    Dim state As RTC_SESSION_STATE
    Dim objSessionEvent As IRTCSessionStateChangeEvent

    On Error GoTo CheckError

    '----- Setzen des lokalen objSessionEvent Objekts auf das übergebene      -----
    '----- pEvent Objekt, um Zugriff auf das IRTCSessionStateChangeEvent In-  -----
    '----- terface zu erhalten. Der Zustand wird lokal in "state" gespeichert.-----

    Set objSessionEvent = pEvent
    state = objSessionEvent.state

    '----- Behandlung der eingehenden Anruf-Meldungen. Es wird jeweils der -----
    '----- aktuelle Zustand ausgegeben und die Buttons des User Interfaces -----
    '----- gesperrt/freigegeben.                                           -----

    Select Case state

        Case RTCSS_IDLE

            lblStatus.Caption = "IDLE"

        Case RTCSS_INCOMING

            '----- Ein eintreffender Anruf wird nur beantworted, wenn keine -----
            '----- andere Sitzung offen ist.                                -----

            If ((g_objSession Is Nothing) Or (g_objSession.state <>
                RTCSS_CONNECTED)) Then
                lblStatus.Caption = "INCOMING"
                cmdTerminate.Enabled = True
                cmdConnect.Enabled = False
                cmdAnswer.Enabled = True

                ' ----- Es ist ein Anruf eingegangen! Läuten des lokalen PCs. -----

                Call g_objRTCClient.PlayRing(RTCRT_PHONE, True)

                ' ----- Die eingehende Sitzung annehmen. -----

                Set g_objSession = objSessionEvent.Session

                ' ----- Automatisch antworten, wenn Auto Answer gesetzt ist. -----

                If chkAutoAnswer.Value = 1 Then
                    Call g_objSession.Answer
                End If
```

```
            End If

        Case RTCSS_ANSWERING

            lblStatus.ForeColor = vbBlue
            lblStatus.Caption = "ANSWERING"

            ' ----- Läuten beenden und Ausgabegerät freigeben. -----

            Call g_objRTCClient.PlayRing(RTCRT_PHONE, False)

            ' ---- Der Answer Button wurde bereits gedrückt, also abschalten. -----

            cmdAnswer.Enabled = False

        Case RTCSS_INPROGRESS

            lblStatus.ForeColor = vbBlue
            lblStatus.Caption = "INPROGRESS"

            '---- Der Wizard darf nur verwendet werden, wenn keine Verbindung -----
            '---- besteht.                                                     -----

            cmdConnect.Enabled = False
            cmdTerminate.Enabled = True
            cmdWizard.Enabled = False

        Case RTCSS_CONNECTED

            lblStatus.ForeColor = vbBlue
            lblStatus.Caption = "CONNECTED"

            '----Im Zustand "connected" wird das KeyPad angezeigt und aktiviert ---

            fraTitel.Visible = False
            fraKeyPad.Visible = True
            Call EnableKeyPad

        Case RTCSS_DISCONNECTED

            '---- Es besteht keine Verbindung mehr, also können die Sitzungs- ----
            '---- und Teilnehmerobjekte freigeben werden.                     ----

            Set g_objSession = Nothing
            Set g_objParticipant = Nothing

            '----- Das KeyPad ist nur im Zustand "connected" aktiviert. -----

            fraTitel.Visible = True
            fraKeyPad.Visible = False
            Call DisableKeyPad

            lblStatus.ForeColor = vbBlue
            lblStatus.Caption = "DISCONNECTED"

            cmdAnswer.Enabled = False
            cmdTerminate.Enabled = False
            cmdConnect.Enabled = True
            cmdWizard.Enabled = True

    End Select

    ' ----- Speicher freigeben -----
```

```
    Set objSessionEvent = Nothing

    Exit Sub

CheckError:
    Call ErrorAusgabe
Resume Next

End Sub

'----- Umschalten zwischen Keypad und Wizard-Button -----

Private Sub fraKeyPad_Click()

            fraTitel.Visible = True
            fraKeyPad.Visible = False

End Sub

'----- Umschalten zwischen Wizard-Button und Keypad -----

Private Sub lblTitel2_Click()

            fraTitel.Visible = False
            fraKeyPad.Visible = True

End Sub

'----- Diese Prozedur wird durch Anklicken des PcToPc OptionButtons aufgerufen ----

Private Sub optPcToPc_Click()

    ' ----- Zeigt die lokale User-URI an und fragt nach der Zieladresse -----

    txtLocalURI.Text = g_objRTCClient.LocalUserURI
    txtEndURI.Text = "Enter destination address!"

    ' ----- Ausgegebener Hinweistext bei längerem Verharren am Objekt -----

    txtEndURI.ToolTipText = "Insert destination SIP-address of the form
                            ""sip:user@host.domain"" or ""sip:user@IP-address"""

End Sub

'--- Diese Prozedur wird durch Anklicken des PcToPhone OptionButtons aufgerufen ---

Private Sub optPcToPhone_Click()

    '--- Zeigt die lokale UserURI an und fragt nach der gewünschten Tel.Nummer ----

    txtLocalURI.Text = g_objRTCClient.LocalUserURI
    txtEndURI.Text = "Enter number to call!"

    ' ----- Ausgegebener Hinweistext bei längerem Verharren am Objekt -----

    txtEndURI.ToolTipText = "Insert destination SIP-address of the form
                           ""sip:+1234567890@gateway.com;user=phone"""

End Sub

'---- Zur einfacheren Eingabe von Adressen wird der String "sip:" vorgegeben ----

Private Sub txtEndURI_Click()
```

```
            If txtEndURI.Text = "Enter destination address!" Or txtEndURI.Text =
                                "Enter number to call!" Then
               txtEndURI.Text = "sip:"
            End If

End Sub

Private Sub cmdWizard_Click()

    On Error GoTo CheckError

    Dim hwndParent As Long

    '----- hwndParent ist ein Handle zu jenem Objekt, auf dem sich das -----
    '----- Steuerelement befindet.                                      -----

    hwndParent = UserControl.hWnd

    ' ----- Aufruf des Tuning Wizards -----

    Call g_objRTCClient.InvokeTuningWizard(hwndParent)

    Exit Sub

CheckError:
    Call ErrorAusgabe
Resume Next

End Sub

Private Sub cmdConnect_Click()

    '----- Deklaration der Lokalen- und der Ziel-URI -----

    Dim strLocalURI As String
    Dim strDestURI As String

    On Error GoTo CheckError

    '----- Um einen Anruf zu tätigen, muss zuerst abhängig von der gewählten -----
    '----- Sitzungsart eine Sitzung mit CreateSession generiert werden.      -----

    If optPcToPc.Value = True Then
       strDestURI = Trim(txtEndURI.Text)
       Set g_objSession = g_objRTCClient.CreateSession(RTCST_PC_TO_PC, vbNull
                                                       String, Nothing, 0)
    ElseIf optPcToPhone = True Then

    '---- Es ist die Sitzungsart PcToPhone ausgewählt. Um von einem PC aus ein ----
    '---- PSTN-Telefon zu erreichen, muss ein Gateway angewählt werden. Die    ----
    '---- eingegebene URI muss also die Form "sip:+123455678978@gateway.com;   ----
    '---- user=phone" haben.                                                   ----

       strDestURI = Trim(txtEndURI.Text)
       Set g_objSession = g_objRTCClient.CreateSession(RTCST_PC_TO_PHONE, vbNull
                                                       String, Nothing, 0)
    End If

    '---- Nachdem die Sitzung generiert wurde, kann ein Teilnehmer hinzugefügt ----
    '---- werden um eine Sitzung zu initiieren.                                ----

    Set g_objParticipant = g_objSession.AddParticipant(strDestURI, "Any Name")

    Exit Sub
```

```
CheckError:
    Call ErrorAusgabe
Resume Next

End Sub

Private Sub cmdTerminate_Click()

    On Error GoTo CheckError

    If g_objSession.state = RTCSS_INCOMING Then

        '----- Einkommenden Anruf zurückweisen -----

        Call g_objSession.Terminate(RTCTR_REJECT)

        '----- Läuten beenden und Audio Übertragungsgerät freigeben. -----

        Call g_objRTCClient.PlayRing(RTCRT_PHONE, False)
    Else

        '----- Beenden der Sitzung -----

        Call g_objSession.Terminate(RTCTR_NORMAL)
    End If
    Exit Sub

CheckError:
    Call ErrorAusgabe
Resume Next

End Sub

Private Sub cmdAnswer_Click()

    On Error GoTo CheckError

    '----- Einen eingehenden Anruf entgegennehmen -----

    Call g_objSession.Answer
    Exit Sub

CheckError:
    Call ErrorAusgabe
Resume Next

End Sub

Private Sub cmdKey_Click(Index As Integer)

   '----- Durch die Verwendung eines Tasten-Feldes ist der Aufruf der SendDTMF ----
   '----- Methode des RTCClient Objekts mit dem richtigen Parameter einfach    ----
   '----- über den Feldindex lösbar.                                           ----

    On Error GoTo CheckError

    Select Case Index
        Case 0
            g_objRTCClient.SendDTMF (RTC_DTMF_STAR)
        Case 1
            g_objRTCClient.SendDTMF (RTC_DTMF_1)
        Case 2
            g_objRTCClient.SendDTMF (RTC_DTMF_2)
```

```
        Case 3
            g_objRTCClient.SendDTMF (RTC_DTMF_3)
        Case 4
            g_objRTCClient.SendDTMF (RTC_DTMF_4)
        Case 5
            g_objRTCClient.SendDTMF (RTC_DTMF_5)
        Case 6
            g_objRTCClient.SendDTMF (RTC_DTMF_6)
        Case 7
            g_objRTCClient.SendDTMF (RTC_DTMF_7)
        Case 8
            g_objRTCClient.SendDTMF (RTC_DTMF_8)
        Case 9
            g_objRTCClient.SendDTMF (RTC_DTMF_9)
        Case 10
            g_objRTCClient.SendDTMF (RTC_DTMF_POUND)
        Case 11
            g_objRTCClient.SendDTMF (RTC_DTMF_0)
    End Select

    Exit Sub

CheckError:
    Call ErrorAusgabe
Resume Next

End Sub

Private Sub DisableKeyPad()

    Dim iIndex As Integer

    '----- Deaktivieren aller Tasten des KeyPads -----

    For iIndex = 0 To 11
        cmdKey(iIndex).Enabled = False
    Next

End Sub


Private Sub EnableKeyPad()

    Dim iIndex As Integer

    '----- Aktivieren aller Tasten des KeyPads -----

    For iIndex = 0 To 11
        cmdKey(iIndex).Enabled = True
    Next

End Sub

Private Sub ErrorAusgabe()

    '----- Diese Prozedur wird bei Auftritt eines Fehlers ausgeführt, um -----
    '----- die Fehlernummer bzw. den zugehöhrigen Text auszugeben.       -----

    lblStatus.ForeColor = vbRed
    lblStatus.Caption = Err.Description
    Err.Clear

End Sub
```

Literaturverzeichnis

[1] APPLEMAN D.: „Dan Developing COM/ActiveX Components with Visual Basic 6“; Markt & Technik Verlag; München; 1998. http://www.mut.de

[2] BAYER J.: „Nitty Gritty, Visual Basic 6“; Addison-Wesley Verlag; München; 2000. http://www.nitty-gritty.de

[3] GUREWICH N.: „Teach Yourself Visual Basic 5 in 21 Days“; SAMS Publishing; Indianapolis Indiana; 1997.

[4] MICROSOFT: „Helpfile: Real Time Communications (RTC) Client Application Programming Interface (API)“; Microsoft Corporation 2001-2003. http://download.microsoft.com/download/1/e/5/1e55f794-0a9a-43c4-8698-72e04c97ff72/RtcApiSdk.msi