برنامه نویسی شبکه و اینترنت در VB بخش اول

مروری بر TCP/IP

نکته : مطالب زیر تنها در حد یک یادآوري مي باشد . اگر اطلاعات کمي در مورد TCP/IP دارید به کتابهاي موجود مراجعه کنید .

پروتکل Protocol : قراردادي است براي برقراري ارتباط در شبکه

مدل TCP/IP : مجموعه اي از پروتکلهاي ارتباطي مرتبط بهم است که مکانیزمها و سرویسهاي مورد نیاز جهت برقراري ارتباط در اینترنت را مهیا مي کنند . این مدل شامل ۴ لایه است :
۱ - لایه کاربرد Application Layer : شامل برنامه هاي کاربردي و پروتکلهایي مثل Http، Ftp، Smtp، Pop و Telnet مي باشد .
۲ - لایه انتقال Transport Layer : این لایه شامل دو پروتکل TCP و UDP است . پروتکل TCP وظیفه کنترل رسیدن بسته هاي داده به مقصد (TCP/IP داده ها را به بسته هاي کوچکي تقسیم مي کند که هر بسته حاوي آدرس فرستنده ، گیرنده و شماره بسته مي باشد) ، تصحیح خطا و مرتب سازي بسته ها را برعهده دارد . UDP پروتکلي شبیه TCP است با این تفاوت که هیچ ضمانتي براي رسیدن بسته هاي اطلاعاتي در آن وجود ندارد و معمولاً در انتقال صوت و ویدئو روي اینتزنت استفاده مي شود .
۳ - لایه اینترنت Layer Internet : شامل پروتکل IP است که مسئول مسیریابي بسته هاي اطلاعاتي مي باشد .
۴ - لایه دسترسي به شبکه Link Layer : شامل بخشي از هسته سیستم عامل و نیز درایورهاي واسط شبکه براي کار با سخت افزار شبکه مي باشد .

سوکت Socket و پورت Port : سوکت یک ورودي انتزاعي در لایه انتقال مي باشد که براي ایجاد ارتباطات مختلف TCP/IP بکار مي رود . اغلب برنامه هاي کاربردي که از TCP و UDP استفاده مي کنند ، عملیات انتقال اطلاعات خود را با ساخت یک سوکت و سپس انجام یکسري عملیات روي آن انجام مي دهند . این عملیات عبارتند از :
۱ - عملیات کنترلي : شامل اختصاص یک شماره پورت به سوکت ، initiate کردن یا accpet کردن یک ارتباط ، از بین بردن سوکت
۲ - عملیات انتقال داده : شامل نوشتن داده روي سوکت و خواندن داده از سوکت
۳ - عملیات بررسي وضعیت : مثل پیدا کردن آدرس IP مربوط به سوکت ، پیدا کردن شماره پورت سوکت و غیره

HTTP : پروتکل انتقال داده براي وب است .

FTP : پروتکل انتقال فایل روي اینترنت است .

SMTP و POP : پروتکلهاي ارسال و دریافت email مي باشند .

کنترل Web Browser – ساخت مرورگر صفحات وب

برنامه Internet Explorer يا iexplore.exe در واقع برنامه كوچكي است كه وظيفه اصلي آن ايجاد چارچوبي براي بهم پيوستن عناصر مختلف است و اين عناصر هستند كه وظايف اصلي مثل load كردن صفحات وب ، اجراي كدهاي Html و غيره را انجام مي دهند . اصلي ترين عنصري كه مستقيماً توسط iexplore.exe استفاده مي شود كنترل Webbrowser (موجود در فايل shdocrw.dll) مي باشد . وظيفه اين فايل dll ، عبارت است از حركت بين صفحات وب ، مديريت تاريخچه صفحات ديده شده و غيره . اين فايل خود از فايل ديگري بنام Mshtml.dll استفاده مي كند كه وظيفه آن بررسي و اجراي فايلهاي html است . مايكروسافت به برنامه نويسان اين امكان را داده كه بتوانند در برنامه هايشان از كنترل webbrowser استفاده كنند . با استفاده از اين كنترل مي توان به سادگي يک مرورگر وب تقريباً كامل ساخت .

خصوصيات كنترل Webbrowser :
Webbrowser علاوه بر خواص استانداردي مثل width، height و ... خواص زير را دارد :
1 – Busy : اگر در حال load كردن يک صفحه يا در حال جستجو در وب باشد اين خاصيت True است . توسط متد Stop مي توان عمليات جاري را متوقف كرد .
2 – Container : ارجاع به شي نگهدارنده كنترل webbrowser
3 – Document : ارجاع به صفحه html فعلي . براي كار با اين صفحه html مي توان از خواص و متدهابي شي Document استفاده كرد .
4 – LocationName : حاوي آدرس محلي است كه اكنون در كنترل webbrowser، load شده است . اگر اين محل يک صفحه html باشد عنوان آن صفحه خواهد بود و اگر اين محل يک فايل در شبكه باشد مسير كامل آن فايل خواهد بود .
5 – LocationURL : حاوي url محلي است كه فعلاً در كنترل webbrowser، load شده است .
6 – Offline : اگر كنترل webbrowser در حالت عدم اتصال باشد مقدار آن True و در غيراينصورت False است .
7 – Parent : فرمي را نشان مي دهد كه كنترل webbrowser در آن قرار دارد .
8 – ReadyState : وضعيت كنترل webbrowser را برمي گرداند .

متدهاي كنترل webbrowser : اين متدها مربوط به مرور در صفحات وب هستند :
1- GoBack : در ليست تاريخچه url ها ، يكي به عقب برمي گردد .
2 – GoForward : در ليست تاريخچه url ها ، يكي به جلو مي رود .
3 – GoHome : به homepage مرورگر مي رود .
4 – Navigate : به يک url يا فايل مي رود . ساختار اين متد بصورت زير است :

Flags,][TargetFrameName,][PostData,][Headers]x] Navigate URL

URL آدرس مقصد مي باشد . Flags نحوه باز شدن آدرس مقصد را تعيين مي كند . اگر اين پارامتر ذكر نشود آدرس جديد در پنجره فعلي باز خواهد شد و به ليست تاريخچه اضافه شده و اگر كپي آن در cache temperory موجود باشد از آنجا خوانده مي شود . مقادير پارامتر Flags عبارتند از :
- NavOpenInNewWindow : آدرس جديد را در پنجره جديدي باز مي كند .
- NavNoHistory : به ليست تاريخچه اضافه نمي شود بلكه جايگزين صفحه فعلي مي شود .
- NavNoReadFromCache : صفحه جديد از cache خوانده نمي شود .
- NavNoWriteToCache : صفحه جديد روي cache نوشته نمي شود

Event هاي كنترل webbrowser : اين event ها مربوط به مرور در وب و تغيير حالت آن هستند :
1 – CommandStateChange : براي فعال يا غيرفعال كردن دكمه هاي Forward و Back در مرورگر استفاده مي شود .
شكل كلي فراخواني اين event بصورت زير است :

(Command As Long, ByVal Enable As Boolean Private Sub WebBrowser1_CommandStateChange(ByVal

که command فرماني است که حالت فعال آن تغيير کرده است و دو مقدار مي گيرد : 1 و 3 که بترتيب معادل فرمانهاي GoForward و GoBack هستند .
Enable فعال يا غيرفعال بودن فرمان را تعيين مي کند .
2 – DocumentComplete : اين event زماني فعال مي شود که صفحه در حال load شدن به حالت ReadyState_Complete برود . شکل کلي فراخواني اين event بصورت زير است :
(Variant Private Sub WebBrowser1_DocumentComplete(ByVal pDisp As Object, URL As
که pDisp ارجاعي به کنترل webbrowser است که event در آن رخ داده است و URL آدرس صفحه در حال load شدن است .
3 – DownloadBegin : اين event در آغاز حرکت به صفحه جديد روي مي دهد و هيچ پارامتري نمي گيرد . مرورگر مي تواند در اين event پيغامي براي شروع عمليات جديد نشان مي دهد .
4 – DownloadComplete : اين event در پايان عمليات يا در صورت انصراف کاربر يا بروز خطا روي مي دهد .
5 – ProgressChange : با بروز هر تغييري در وضعيت load ، اين event روي مي دهد . شکل کلي فراخواني آن بصورت زير است :

(Progress As Long, ByVal ProgressMax As Long Private Sub WebBrowser1_ProgressChange(ByVal

که Progress نشان دهنده پيشرفت عمليات (بايتهاي load شده) است . پارامتر ProgressMax تعداد کل بايتهايي که بايد load شوند را نشان مي دهد بنابر اين :

درصد پيشرفت عمليات load=Progress/ProgressMax)*100)

يک مثال ساده :
از منوي project مورد components را انتخاب کنيد و از ليست کنترلها ، Microsoft Internet Controls را به toolbar خود اضافه کنيد . يک کنترل WebBrowser روي فرم قرار دهيد و سايز آنرا به اندازه ابعاد فرم خود قرار دهيد . يک textbox و يک دکمه روي فرم قرار دهيد . کد زير را براي event مربوط به کليک دکمه بنويسيد :

```
WebBrowser.Navigate textbox.text
```

کنترل Internet Transfer - قسمت اول

مقدمه : کنترل Internet Transfer نسبت به کنترل WebBrowser که در روزهاي قبلي معرفي شد در سطح پايينتري قرار دارد . اين کنترل با استفاده از دو پروتکل HTTP و FTP مي تواند داده ها را منتقل کند . اين کنترل زمانيکه از پروتکل HTTP استفاده مي کند با همان روش کنترل WebBrowser به سرويس دهنده صفحات وب متصل مي شود اما بجاي آنکه صفحه وب را نمايش دهد متن Html صفحه را بازيابي مي کند . همچنين زمانيکه اين کنترل از پروتکل FTP استفاده مي کند قادرست فايلها را بين کامپيوترهاي روي شبکه منتقل سازد .
اتصالات HTTP : همانطور که مي دانيد ، پروتکل HTTP استاندارد وب مي باشد . صفحات وب با زبان Html نوشته مي

شوند و انتقال آنها از server به client توسط پروتكل HTTP صورت مي گيرد .
متد OpenURL: ساده ترين راه استفاده از كنترل IT متد OpenURL است . شكل كلي اين متد بصورت زير است :

Inet.OpenURL(url,DataType)x

كه url آدرس صفحه وب و DataType نوع داده بازيابي شونده است و دو مقدار icString (داده متني) يا icByteArray (داده باينري) را مي گيرد . مقدار بازگشتي اين متد ، داده هاي منتقل شده است .
اين متد بصورت سنكرون كار مي كند يعني در تمام مدت كار آن برنامه نمي تواند كار ديگري انجام دهد . اگر از icByteArray استفاده كنيد بايد مقدار بازگشتي آنرا در يک ارايه بايت قرار دهيد .
مثال 1 : از بخش Component در منوي Project مورد Microsoft Internet Transfer Control 6.0 را به toolbar خود اضافه كنيد . سپس يک كنترل IT روي فرم قرار دهيد و همچنين يک Rich Textbox و يک دكمه روي فرم قرار دهيد و كد زير را براي event مربوط به كليک دكمه بنويسيد :

TextBox.text=Inet.OpenURL("www.microsoft.com",icString)x

مثال 2 : كد زير داده هاي باينري را از اينترنت خوانده و آنها را در يک فايل ذخيره مي كند :

byte Dim b() as
B()=Inet.OpenURL(ftp://ftp.microsoft.com/test.zip,icByteArray)x
test.zip" For Access Write As #1\" & App.path Open
Put #1,b()x
1# Close

رويداد StateChanged : كنترل IT فقط يک event دارد كه StateChanged مي باشد . اين event زماني روي مي دهد كه State كنترل تغيير كند . State هر اتفاقي است كه برنامه بايد از آن مطلع شود . تعريف كلي اين event بصورت زير است :

Inet_StateChanged(ByVal NewState As Integer)x

كه NewState مقداري است كه حالت جديد را بيان مي كند . مقادير ممكن اين پارامتر عبارتند از :
icNone : حالت تغيير نكرده است .
icResolvingHost : در حال جستجوي آدرس IP كامپيوتر موردنظر .
icHostResolved : آدرس IP كامپيوتر موردنظر يافت شد .
icConnecting : در حال اتصال به كامپيوتر مقصد
icConnected : اتصال به كامپيوتر مقصد برقرار شد .
icRequesting : در حال ارسال درخواست به كامپيوتر مقصد
icRequestSent : درخواست به كامپيوتر مقصد ارسال شد .
icReceivingResponse : در حال دريافت پاسخ از كامپوتر مقصد .
icResponseReceived : پاسخ كامپيوتر مقصد دريافت شد .
icDisconnecting : در حال قطع اتصال با كامپيوتر مقصد .
icDisconnected : اتصال مقصد با موفقيت قطع شد .
icError : در ارتباط با كامپيوتر مقصد خطايي رخ داده است .
icResponseCompleted : تكميل پاسخ – تمام داده ها دريافت شد .
تشخيص خطا در عمليات انتقال داده اهميت بالايي دارد و StateChanged در صورت بروز هر خطايي مقدار icError را برمي گرداند و اطلاعات خطا را در دو خاصيت ResponseCode و ResponseInfo برمي گرداند .
انتقال داده بصورت آسنكرون: كنترل IT متدهاي انعطاف پذير ديگري هم دارد كه آسنكرون هستند و اجازه مي دهند تا همزمان با عمليات انتقال داده ، برنامه به وظايف ديگري هم بپردازد . اين متدها با استفاده از Event Driven Model كار مي كنند . بدين معني كه وقتي برنامه درخواست انتقال داده اي را مي دهد كنترل IT درخواست را در زمينه برنامه انجام

مي دهد و برنامه ازاد است تا به كارهاي ديگرش بپردازد . زمانيكه داده ها بازيابي شود ، داده ها را از بافر داخلي كنترل IT مي خواند .
متد GetChunk : در عمليات انتقال آسنكرون ، بايستي داده را توسط اين متد از بافر داخلي كنترل IT بگيريم :

```
Inet.GetChunk(datasize[,datatype])x
```

كه پارامتر datasize از نوع long بوده و تعيين مي كند چند بايت از بافر خوانده شود و پارامتر اختياري datatype نوع داده را مشخص مي كند و مي تواند مقادير icString و icByteArray را بگيرد .
زمانيكه StateChanged وارد حالتهاي icResponseReceived و يا icResponseCompleted شد بايد از GetChunk استفاده كنيد . بدين صورت كه از يک حلقه استفاده مي كنيم تا كل بافر را بخوانيم :

```
Integer)x Private Sub Inet_StateChanged(Byval State as
Dim temp1,temp2
Select Case State
icResponseCompleted Case
temp1=””x
temp2=””x
Do
temp1=Inet.GetChunk(512,icString)x
temp1 & temp2=temp2
Loop Until temp1=””x
End Select
End Sub
```

براي بالابردن كارايي ، بهتر است از قطعات كوچک (بين 512 تا 1024 بايتي) استفاده كنيد .
متد Execute : و اما انعطاف پذيرترين متد كنترل IT ، متد Execute است . فرمت كلي اين متد بصورت زير است :

```
Inet.Execute(url,Command,Data,RequestHeaders)x
```

كه url آدرس مقصد ، Command فرماني است كه به كامپيوتر مقصد داده مي شود و Data و RequestHeaders اطلاعات اضافي لازم براي اجراي فرمان داده شده است . فرمانهاي Command همان فرمانهاي HTTP هستند كه عبارتند از :
- GET : دريافت داده ها از كامپيوتر مقصد
- HEAD : دريافت اطلاعات header از كامپيوتر مقصد
- POST : ارسال اطلاعات لازم براي تكميل درخواست
- PUT : ارسال فايل براي كامپيوتر ميزبان (upload)
فرمان GET پركاربردترين فرمان متد Execute است و داده هاي خوانده شده را در بافر داخلي بافر كنترل IT قرار مي دهد تا بتوان با متد GetChunk آنها را بازيابي نمود .
مثال :

```
Inet.Execute http://www.microsoft.com,”GET”x
```

ساير خواص كنترلIT :
- AccessType : نوع دسترسي كنترل IT به اينترنت را مشخص مي كند و سه مقدار مي تواند بگيرد :
icUseDefault : استفاده از تنظيمات رجيستري براي دسترسي به اينترنت
icDirect : اتصال مستقيم كنترل IT به اينترنت
icNamedProxy : اتصال به اينترنت توسط پروكسي
- Document : نام صفحه پيش فرض كه در متد Execute از آن استفاده مي شود . اگر به متد Execute پارامتر url را ندهيد از اين صفحه پيش فرض استفاده مي كند .

- Password : کلمه رمز عبور کامپيوتر ميزبان FTP
- Procotol : نوع پروتکل مورد استفاده در متد Execute را مشخص مي کند و 5 مقدار مي تواند بگيرد :
icUnknown : نامعلوم
icDefault : پروتکل پيش فرض
icFTP : پروتکل FTP
icHTTP : پروتکل HTTP
icHTTP : پروتکل حفاظت شده HTTP
- Proxy : نام ميزبان پروکسي
- RequestTimeOut : مدت زماني که کنترل IT صبر مي کند تا اطلاعات را دريافت کند . اگر اين خاصيت صفر باشد کنترل تا هر زمان که لازم باشد براي دريافت پاسخ صبر مي کند . در حالت سنکرون (متد OpenURL) بعد از سپري شدن اين مدت زمان ، يک خطا توليد مي شود و در حالت آسنکرون (متد Execute) رويداد StateChanged مقدار خطا را بر مي گرداند
- ResponseCode : بعد از بروز حالت icError اين خاصيت کد خطا را مي دهد .
- ResponseInfo : توضيحي درباره خطا
- StillExecuting : اگر True باشد يعني کنترل مشغول انجام کار است .
- URL : آدرس مقصد در متدهاي OpenURL و يا Execute
- UserName : نام کاربر براي ورود به کامپيوتر ميزبان FTP

کنترل Internet Transfer - قسمت دوم

اتصالات FTP

پروتکل FTP علاوه بر نقل و انتقال فايل بين دو کامپيوتر ، مي تواند نوعي مديريت فايل (مثل حذف فايل يا ايجاد پوشه) روي کامپيوتر مقصد را انجام دهد . FTP در انتقال فايل بسيار قويتر از HTTP است ولي به مراتب پيچيده تر از HTTP مي باشد اما کنترل IT اين پيچيدگيها را از ديد برنامه نويس مخفي کرده است .
براي کار با سرورهاي FTP بايد به آنها Login نمود . نوع خاصي از Login به نام Anonymous Login (ورود ناشناس) وجود دارد که با آن کاربران مي توانند بدون محدوديت از سايت FTP استفاده کنند . توجه کنيد که حتي براي ورود ناشناس هم نياز به نام کاربر و کلمه عبور است . براي ارسال نام کاربر و کلمه عبور از خواص username و password کنترل IT استفاده مي شود . اگر خاصيت username خالي باشد (blank) ، کنترل IT بطور خودکار از anonymous استفاده مي کند و آدرس email کاربر بعنوان passowrd استفاده مي شود .
استفاده از متد OpenURL: متد OpenURL ساده ترين راه انجام عمليات FTP است . دستور زير از يک سايت FTP ليست مي گيرد :

```
Text.text=Inet.OpenURL("ftp://ftp.microsoft.com",icString)x
```

براي خواندن فايل از يک سايت FTP بايد در حالت باينري کار کرد :

```
b()=Inet.OpenURL("ftp://ftp.microsft.com/test.zip",icByteArray)x
```

استفاده از متد Execute: متد Execute قابليتهاي بيشتري دارد و اجرای آن در FTP نياز به دو پارامتر دارد :

```
Inet.Execute(url,operation)x
```

که url آدرس سايت FTP بهمراه نام و مسير فايل و پارامتر operation يک فرمان FTP است . کنترل IT با داده هاي خوانده شده FTP به دو طريق رفتار مي کند :
برخي از داده ها مثل پاسخ فرمان DIR در بافر کنترل IT قرار مي گيرد و بايد آنها را با متد GetChunk خواند .
برخي ديگر از داده ها مثل فايل خوانده شده با فرمان GET مستقيماً روي ديسک نوشته مي شوند و ديگر نيازي به استفاده از متد GetChunk نيست .
فرامين FTP بسيار قوي هستند و حتي به شما اين امکان را مي دهند که فايلها را به روي کامپيوتر مقصد کپي کنيد ، به پوشه هاي کامپيوتر مقصد برويد ، فايلها را حذف کنيد و يا تغيير نام دهيد . البته بايد توجه داشت که فرامين قابل اجرا به نوع ورود به سيستم FTP بستگي دارد . اگر با کاربر anonymous به يک سايت FTP وارد شويد تنها مي تواند فايلها را ببيند و آنها را download کنيد .
مهمترين فرامين FTP عبارتند از :
CD path : به دايرکتوري path مي رويد .
CDUP : به يک دايرکتوري بالاتر مي رود .
CLOSE : بستن اتصال FTP
DELETE file1 : حذف فايل file1
DIR file1 : جستجوي فايل file1 روي دايرکتوري جاري
MKDIR path : ايجاد يک دايرکتوري با نام path
PUT file1 file2 : فايل file1 را از کامپيوتر مبدا روي فايل file2 در کامپيوتر مقصد کپي مي کند .
PWD : نام دايرکتوري جاري در کامپيوتر مقصد را برمي گرداند .
QUIT : قطع اتصال FTP
GET file1 file2 : فايل file1 را از کامپيوتر مقصد روي فايل file2 در کامپيوتر مبدا کپي مي کند .
RENAME file1 file2 : تغيير نام فايل file1 به file2
RMDIR path : حذف دايرکتوري path در کامپيوتر مقصد
SIZE file1 : بدست آوردن تعداد بايتهاي فايل يا دايرکتوري file1
مثال :

```
test.zip")x Inet.Execute("ftp://ftp.microsoft.com","GET
```

کنترل WinSock - قسمت اول

مقدمه :
کنترل WinSock نسبت به تمام کنترلهاي اينترنت در سطح پايينتري قرار دارد . اين کنترل امکان ايجاد سرويسهاي شبکه اي مبتني بر پروتکلهاي TCP و UDP را مهيا مي کند . بعبارت ديگر توسط اين کنترل مي توان برنامه هاي کاربردي Client/Server (سرويس گيرنده / سرويس دهنده) ايجاد و با استفاده از پروتکل TCP و يا UDP بين آنها ارتباط برقرار نمود .
با تنظيم خصوصيات و فراخواني متدهاي اين کنترل مي توانيد به راحتي به يک کامپيوتر راه دور متصل شويد و داده ها را در هر دو جهت جابجا نمائيد . نمونه کاربرهايي که مي توان با اين کنترل ايجاد نمود :
Client-server chat، Mail client، Mail server، Proxy Server، Game Network، Port Scanner ، پياده سازي الگوريتم هاي موازي و ...

مباني TCP :
پروتکل کنترل اينترنت (Transfer Control Protocol) اجازه مي دهد يک اتصال (Connection) را از طريق سوکت (socket) به يک کامپيوتر راه دور (Computer Remote) ساخته و استفاده کنيد . با استفاده از اين اتصال ، هر دو کامپيوتر مي توانند داده ها را بين خودشان انتقال دهند . برقراري ارتباط از طريق TCP همانند صحبت کردن با تلفن است که بايد حتماً اتصالي بين دو کامپيوتر صورت گيرد تا بتوانند با هم ارتباط برقرار کنند .
اگر يک برنامه Client مي سازيد بايستي بدانيد که نام يا آدرس IP کامپيوتر Server چيست (Remote Host IP) و همچنين از طريق چه پورتي مي توانيد به آن متصل شويد (Remote Port) . حال بايستي به آن پورت Connect کنيد .
همچنين اگر يک برنامه Server مي سازيد بايستي پورتي را که روي آن به درخواستها گوش مي دهيد مشخص کنيد (LocalPort) و سپس به پورت گوش دهيد (Listen) .
زمانيکه يک کامپيوتر Client تقاضاي يک اتصال را مي دهد Server اين درخواست را Accept مي کند .
زمانيکه يک اتصال ساخته مي شود ، هر دو کامپيوتر مي توانند داده را فرستاده و دريافت کنند .

مباني UDP :
پروتکل ديتاگرام کاربر (User Datagram Protocol) پروتکلي بدون اتصال (Connectionless) است . برخلاف TCP ، کامپيوترها نياز به برپا کردن يک اتصال ندارند بنابراين يک برنامه مي تواند يک client و يا يک server باشد . برقراري ارتباط در UDP شبيه ارسال نامه از طريق پست است .
براي انتقال داده توسط UDP ابتدا بايد Local Port کامپيوتر Client تنظيم گردد . کامپيوتر Server تنها بايستي RemoteHost را برابر آدرس کامپيوتر Client قرار دهد و همچنين Remote Port را همان Local Port کامپيوتر Client قرار دهد . سپس دو کامپيوتر مي توانند داده ها را بين خود جابجا کنند .

استفاده از کنترل WinSock :

1 – انتخاب پروتکل: در زمان استفاده از کنترل WinSock اولين کاري که بايد انجام دهيد انتخاب يکي از پروتکلهاي TCP يا UDP است . طبيعت برنامه اي که شما مي سازيد نوع پروتکلي را که بايد استفاده کنيد مشخص مي کند . چند سوال زير به شما کمک مي کند که پروتکل مورد نيازتان را انتخاب کنيد :
- آيا برنامه شما در زمانيکه داده فرستاده مي شود يا دريافت مي شود نياز به اطلاعاتي از طرف Server يا Client دارد ؟ اگر چنين است بايستي يک اتصال TCP قبل از ارسال يا دريافت داده ايجاد شود .
- آيا داده بسيار بزرگ است (مثل تصوير يا فايلهاي صوتي) ؟ زمانيکه يک اتصال TCP ساخته مي شود پروتکل TCP اتصال را باقي نگه مي دارد و درستي ارسال داده تضمين شده است . اين اتصال در هر حال به منابع محاسباتي بيشتري نياز دارد و بنابراين پرهزينه تر است .

- آيا داده متناوب ارسال مي شود يا در يک نشست (Session) ارسال خواهد شد ؟ براي مثال اگر شما يک برنامه مي سازيد که کامپترهاي مشخصي را در يک زمان خاص از انجام شدن عملياتي مطلع مي کند پروتکل UDP مناسب تر است . پروتکل UDP همچنين براي ارسال مقادير کوچک داده اي مناست تر مي باشد .
2 – تنظيم پروتکل : براي تنظيم پروتکلي که مي خواهيد در برنامه تان از آن استفاده کنيد در زمان طراحي برنامه خاصيت Protocol کنترل WinSock را برابر sckTCPProtocol و يا sckUDPProtocol قرار دهيد . همچنين مي توانيد پروتکل خود را توسط کد زير تنظيم کنيد :

```
WinSock.Protocol=sckTCPProtocol
```

3 – مشخص کردن نام کامپيوتان : براي اتصال به کامپيوتر راه دور بايستي آدرس IP و يا نام کامپوتر را بدانيد .
نام کامپيوتر در Control Panel/Network/Identification موجود است . در صورتيکه مي خواهيد دو برنامه Client و Server خود را روي يک کامپيوتر تست کنيد از آدرس IP 127.0.0.1 براي هر دو استفاده کنيد اما اگر دو برنامه را روي دو کامپيوتر مجزا در شبکه قرار داده ايد با اجراي دستور ipconfig در DOS Prompt مي توانيد آدرس IP کامپيوتر ها را بدست آوريد .
4 – ايجاد اتصال TCP : در زمان ساخت برنامه اي که از پروتکل TCP استفاده مي کند ابتدا بايد تصميم بگيريد که اين برنامه Client است يا Server . براي ساخت يک برنامه Server بايستي روي يک پورت خاص Listen کنيد . زمانيکه Client تقاضاي يک اتصال را مي دهد ، برنامه Server مي تواند آنرا Accept کند و بنابراين اتصال کامل شده است . حال Client و Server مي توانند با هم ارتباط داشته باشند .
مراحل زير ساخت يک سرور چت ساده بر مبناي TCP را نشان مي دهد :
- از منوي Project گزينه Components را انتخاب کنيد و در ليست Component ها مورد Microsoft WinSock 6.0 را انتخاب کنيد .
- يک کنترل WinSock در فرم خود قرار دهيد و نام آنرا tcpserver بگذاريد
- دو textbox با نامهاي txtSendData و txtReceiveData و نيز يک دکمه در فرم قرار دهيد .
- کد زير را در رويداد Form_Load بنويسيد :

```
Tcpserver.LocalPort=1000
tcpserver.Listen
```

- زمانيکه درخواستي از طرف Client مي آيد رويداد ConnectionRequest اجرا مي شود . در اين رويداد ابتدا بايد چک کنيد که حالت کنترل بسته باشد . اگر چنين نيست اتصال را قبل از پذيرفتن اتصال جديد ببنديد . سپس تقاضا را بر اساس پارامتر requestID مي پذيريم :

```
(requestID As Long Private Sub tcpserver_ConnectionRequest(ByVal
tcpserver.Close sckClosed Then <> If tcpserver.State
tcpserver.Accept requestID
End Sub
```

- حال اتصال بين Client و Server برقرار شده است . کد زير را براي event مربوط به کليک دکمه Send بنويسيد :

```
Tcpserver.SendData txtSendData.text
```

- اگر داده اي از طرف Client بيايد رويداد DataArrival اجرا مي شود . کد زير را براي اين رويداد بنويسيد :

```
(As Long Private Sub tcpserver_DataArrival(ByVal bytesTotal
Dim strData As String
strData tcpserver.GetData
```

```
txtReceiveData.Text = strData
End Sub
```

- کد زير را براي رويداد Form_Unload بنويسيد :

```
Tcpserver.Close
```

مراحل ساخت يک TCP Client بصورت زير است :
- يک کنترل WinSock در فرم قرار دهيد و نام آنرا tcpclient بگذاريد .
- دو textbox با نامهاي txtsend و txtreceive و نيز يک دکمه با نام sendدر فرم قرار دهيد .
- يک دکمه با نام connect در فرم قرار دهيد .
- کد زير را براي متد Form_Load بنويسيد :

```
tcpclient.RemoteHost=”yourservername”x
tcpclient.RemotePort=1000
```

- کد زير را براي رويداد کليک شدن دکمه connect بنويسيد :

```
tcpclient.Connect
```

- کد زير را براي رويداد کليک شدن دکمه send بنويسيد :

```
tctclient.SendData txtsend.Text
```

- کد زير را براي رويداد DataArrival بنويسيد :

```
(As Long Private Sub tcpclient_DataArrival(ByVal bytesTotal
Dim strData As String
strData tcpclient.GetData
txtreceive.Text = strData
End Sub
```

- کد زير را باري رويداد Form_Unload بنويسيد :

```
Tcpclient.Close
```

کدهاي فوق يک سيستم Client-Server ساده را نشان مي دهد . فايل exe هر دو برنامه را بسازيد و آنها را اجرا کنيد تا بتوانيد سيستم خود را تست کنيد .
5 – پذيرفتن بيش از يک تقاضاي اتصال : Server اي که در بالا ساخته شد تنها مي تواند تقاضاي يک اتصال را بپذيرد . با استفاده از ايجاد يک آرايه از کنترل WinSock مي توان چندين تقاضاي اتصال را پذيرفت . براي اينکار کافي است يک کپي (instance) از کنترل بسازيم (با تنظيم خاصيت Index) و متد Accept را براي instance جديد بکار ببريم . فرض کنيد يک کنترل WinSock با نام sckServer در فرم داريم که خاصيت Index آنرا صفر قرار داده ايم . همچنين يک متغير intMax از نوع Long تعريف مي کنيم که تعداد اتصالات همزمان به Server را نگه مي دارد . در event مربوط به Form_Load کد زير را بنويسيد :

```
intMax=0
sckServer(0).LocalPort=1000
sckServer(0).Listen
```

هر بار که تقاضاي يک اتصال مي رسد کد ابتدا تست مي کند که مقدار Index چقدر است . اگر مقدار Index صفر باشد متغير intMax يکي افزايش مي يابد و از intMax براي ساخت يک instance جديد از کنترل استفاده مي شود . حال از اين instance براي پذيرفتن تقاضاي اتصال استفاده مي گردد . براي اينکار کد زير را براي رويداد ConnectionRequest بنويسيد :

```
(Integer, ByVal requestID As Long Private Sub sckServer_ConnectionRequest(Index As
If Index = 0 Then
1 + intmax = intmax
Load sckServer(intmax)x
0 = sckServer(intmax).LocalPort
sckServer(Index).Accept requestID
End If
End Sub
```

6 – ايجاد اتصال UDP : ساخت يک برنامه UDP ساده تر از برنامه هاي TCP است زيرا پروتکل UDP به اتصال نياز ندارد . در برنامه TCP بالا يک کنترل WinSock بايستي حتماً Listen مي کرد و يک کنترل ديگر يک اتصال را توسط متد Connect ايجاد نمود . در عوض پروتکل UDP نيازي به اتصال ندارد . براي ارسال داده بين دو کنترل WinSock سه مرحله بايستي انجام شود :

- پارامتر RemoteHost برابر نام کامپيوتر مقابل است .
- پارامتر RemotePort برابر پارامتر LocalPort کامپيوتر مقابل
- استفاده از متد Bind براي مشخص کردن LocalPort

چون هر دو کامپيوتر از نظر ارتباط مساوي هستند ، اين نوع برنامه ها را Peer-to-Peer گويند . براي نمونه از کد زير براي ساخت يک برنامه chat استفاده مي کنيم :

- يک کنترل WinSock در فرم قرار دهيد و نام آنرا udppeerA بگذاريد .
- خاصيت Protocol آنرا UDPProtocol قرار دهيد .
- دو textbox با نامهاي txtsend و txtreceive و نيز يک دکمه در فرم قرار دهيد .
- کد زير را براي متد Form_Load بنويسيد :

```
udppeerA.RemoteHost=”nameofpeerB”x
udppeerA.RemotePort=1001
1002 udppeerA.Bind
```

- کد زير را براي event مربوط به کليک دکمه بنويسيد :

```
udppeerA.SendData txtsend.text
```

- کد زير را براي رويداد DataArrival بنويسيد :

```
Dim strData as String
strData udppeerA.GetData
txtreceive.Text=strData
```

براي ساخت UDP peerB مشابه مراحل بالا عمل كنيد فقط خاصيت RemoteHost آنرا نام كامپيوتر PeerA و خاصيت RemotePort آنرا 1002 و خاصيت Bind آنرا 1001 قرار دهيد .

كنترلWinSock - قسمت دوم

بررسی خواص کنترل WinSock :
ByteReceived : مقدار داده دريافت شده (موجود در بافر receive) را نشان مي دهد . توسط متد GetData مي توان اين داده را دريافت نمود .
LocalHostName : نام ماشين محلي را نشان مي دهد . اين پارامتر فقط خواندني است .
LocalIP : آدرس IP ماشين محلي را بصورت يک string برمي گرداند . اين پارامتر فقط خواندني است .
LocalPort : براي خواندن و يا تنظيم شماره پورت محلي بكار مي رود .
Protocol : براي خواندن و يا تنظيم پروتوكل مورد استفاده توسط كنترل WinSock بكار مي رود .
RemoteHost : براي خواندن و يا تنظيم نام يا آدرس IP ماشين راه دور بكار مي رود .
RemoteHostIP : آدرس IP ماشين راه دور را برمي گرداند :
۱- براي برنامه هاي Client بعد از زمانيكه يک اتصال توسط متد Connect پذيرفته شد ، اين خاصيت حاوي آدرس IP ماشين راه دور است .
۲ - براي برنامه Server ، بعد از آمدن يک Connection Request اين خاصيت شامل آدرس IP ماشين راه دور است .
۳ - در زمان استفاده از پروتكل UDP بعد از اينكه رويداد Arrival Data رخ داد اين خاصيت حاوي آدرس IP ماشيني است كه داده را فرستاده .
RemotePort : براي خواندن و يا تنظيم شماره پورت ماشين راه دوري كه مي خواهيد به آن متصل شويد بكار مي رود .
SocketHandle : مقداري را برمي گرداند كه مرتبط با سوكتي است كه كنترل WinSock را مديريت مي كند و براي ارتباط با لايه WinSock بكار مي رود . اين پارامتر فقط خواندني است و تنها براي ارسال به API هاي WinSock طراحي شده است .
State : وضعيت كنترل WinSock را نشان مي دهد . وضعيتهاي ممكن براي State عبارتند از :
۱ - sckClosed : اتصال بسته است .
۲ - sckOpen : اتصال باز است .
۳ - sckListening : حالت گوش دادن به پورت
4 - sckConnectionPending : معلق شدن اتصال
۵ - sckResolvingHost : تصميم گيري در مورد ميزبان
۶ - sckHostResolved : در مورد ميزبان تصميم گيري شد .
۷ - sckConnecting : حالت برقراري ارتباط
۸ - sckConnected : ارتباط برقرار شد .
۹ - sckClosing : حالت قطع اتصال
۱۰ - sckError : حالت خطا

بررسی متدهای کنترل WinSock :
متد Accept : تنها براي برنامه هاي TCP Server بكار مي رود . اين متد براي پذيرفتن يک اتصال در زمان مديريت رويداد ConnectionRequest استفاده مي شود .
متد Bind : اين پارامتر LocalPort و LocalIP يک اتصال را مشخص مي كند .
متد Close : براي بستن يک اتصال TCP و يا بستن يک listening socket بكار مي رود .
متد GetData : بلوک جاري داده دريافت شده را گرفته و آنرا در متغيري از نوع Variant ذخيره مي كند . شكل كلي اين متد بصورت زير است :

WinSock.GetData data[,type][,maxlen]x

که data داده دريافتي است . اگر داده کافي موجود نباشد data برابر empty خواهد بود .
type نوع داده دريافتي است که مي تواند مقادير زير باشد :
vbString - - vbInteger - vbLong - vbSingle - vbDouble - vbDate - vbBoolean - vbError - vbByte
vbArray+vbByte
maxlen حداکثر سايز را در زمان دريافت يک byte Array و يا يک string مشخص مي کند .
متد Getdata در رويداد Data Arrival استفاده مي شود که اين رويداد يک پارامتر با نام TotalBytes دارد . اگر maxlen اي که شما تعيين کرده ايد کمتر از TotalBytes باشد پيغام هشدار شماره ۱۰۰۴۰ دريافت مي کنيد بدين معني که بايتهاي باقيمانده گم خواهند شد .
متد Listen : يک سوکت مي سازد و آنرا در حالت Listen قرار مي دهد . اين متد تنها در اتصالات TCP بکار ميرود .
متد PeekData : مشابه GetData است با اين تفاوت که داده را از صف ورودي حذف نمي کند . اين متد تنها براي اتصالات TCP بکار مي رود .
متد SendData : براي ارسال داده به کامپيوتر راه دور بکار مي رود .

بررسي event هاي کنترل WinSock :

رويداد Close : زماني رخ مي دهد که کامپيوتر راه دور اتصال را ببندد .
رويداد Connect : بعد از اينکه يک اتصال به Server ايجاد شد روي مي دهد . شکل کلي آن بصورت زير است :

```
Private Sub WinSock_Connect(ErrorOccurred As Boolean)x
```

که پارامتر ErrorOccurred دو مقدار دارد : اگر True باشد يعني اتصال Fail شده است و اگر False باشد يعني اتصال با موفقيت انجام شده است .
با رويداد Connect مي توانيد error هايي که در زمان فرايند باز کردن اتصال برگردانده شده را چک کنيد .
رويداد ConnectionRequest : زماني رخ مي دهد که يک کامپيوتر راه دور تقاضاي يک اتصال را بدهد . اين رويداد فقط براي برنامه هاي TCP Server بکار مي رود .
رويداد DataArrival : زماني رخ مي دهد که داده جديدي بيايد .
رويداد Error : زماني رخ مي دهد که يک خطا در فرايند ارتباط رخ دهد (مثلاً Failed to Connect و يا Failed to Send) .
شکل کلي آن بصورت زير است :

```
Private WinSock_Error(number as Integer,description as String,scode as Long,source as String,helpfile as String,helpcontext as Long,canceldisplay as Boolean)x
```

number شماره کد خطا است .
description توضيحي در مورد خطا است .
source توصيف منبع خطا
canceldisplay : مشخص مي کند آيا پيغام خطاي پيش فرض نشان داده شود يا نه
رويداد SendComplete : زماني رخ مي دهد که يک عمل Send تکميل شده باشد .
رويداد SendProgress : زماني رخ مي دهد که کنترل شروع به ارسال داده نمايد . شکل کلي آن بصورت زير است :

```
Private WinSock_SendProgress (bytesSent As Long, bytesRemaining As Long)x
```

که bytesSent تعداد بايتهاي ارسال شده و bytesRemaining تعداد بايتهاي باقيمانده است .

TAPI در ویژوال بیسیک - مقدمه

TAPI چیست ؟

TAPI یا Telephony API یک کتابخانه استاندارد برای کار با مودم و نوشتن برنامه های تلفنی می باشد . برای نمونه می توان از برنامه های Phone Dialer (شماره گیر تلفن) ، برنامه شبکه سازی تلفنی (Dialup Networking) ، برنامه تشخیص پالس مودم برای ضبط اطلاعات وارد شده از طرف کاربران و کاربردهای دیگر در این زمینه نام برد . این کتابخانه به شما کمک کمک می کند تا بدون درگیر شدن با برنامه نویسی سخت افزار مودم و درایور آن بطور مستقیم بتوانید برنامه های کاربردیي در این زمینه بنویسید .

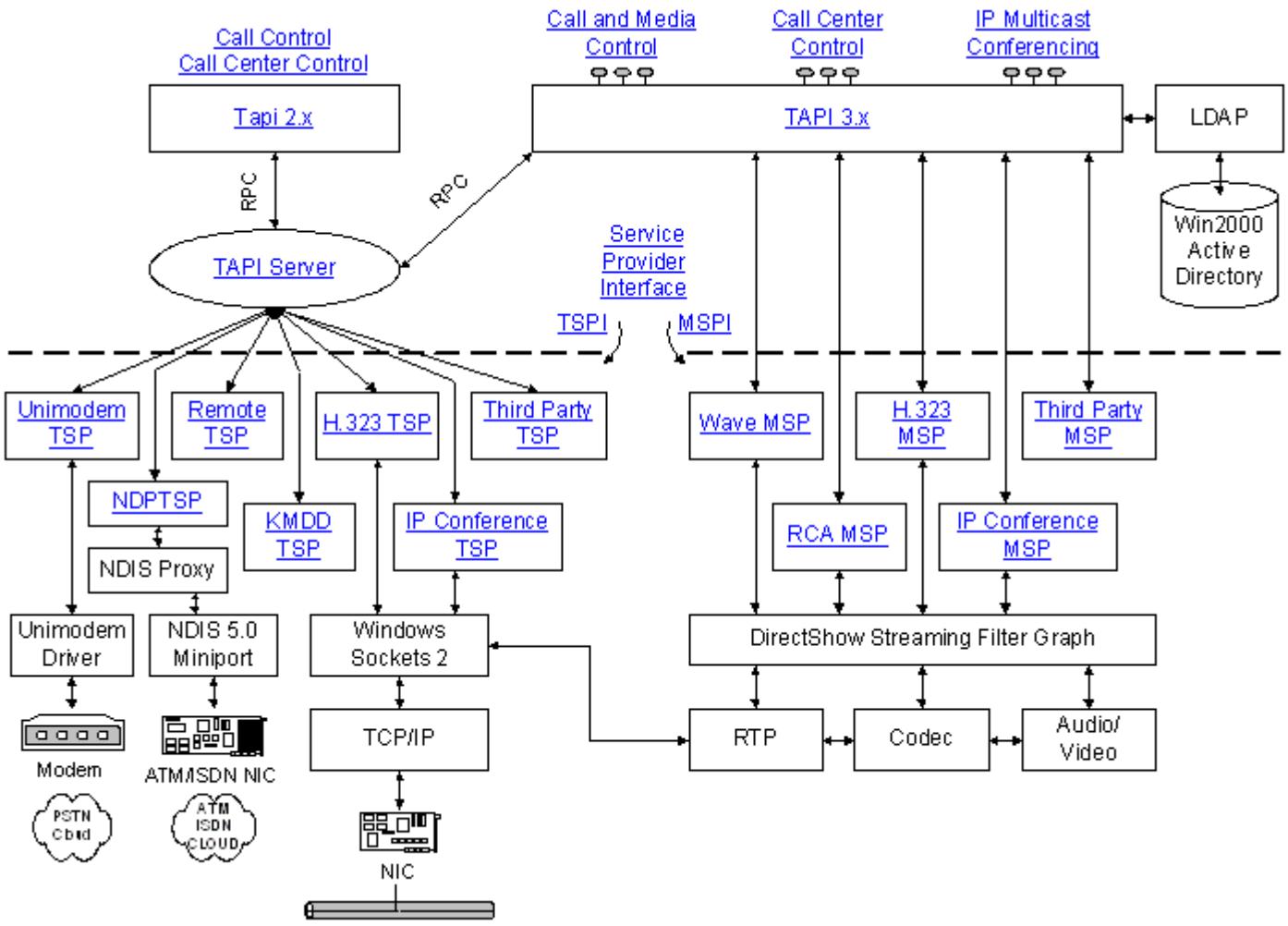
TAPI در ویژوال بیسیک - بخش اول

مروری بر Microsoft Telephony :

Telephony امکان مجتمع سازی کامپیوترها با دستگاههای ارتباطی و شبکه ها را فراهم نموده است . معمولاً دستگاه ارتباطی یک مودم و خط ارتباطی نیز شبکه PSTN (شبکه عمومی تلفن سوئیچینگ) می باشد . برخی از کاربردهای Telephony عبارتند از :

۱ - کنفرانسهای مالتی مدیا بصورت Multicast
۲ - VoIP
۳ - مرکز پاسخ گویي اتوماتیک
۴ - تماس تلفنی از طریق کامپیوتر روی شبکه PSTN

دیاگرام زیر معماری Microsoft Telephony را نشان می دهد :



برنامه های TAPI :

برای نوشتن برنامه های کاربردی با استفاده از TAPI بایستی ابتدا در مورد سطح سرویسی که می خواهیم ارائه دهیم تصمیم گیری کنیم . برای مثال برای نوشتن یک برنامه شماره گیر تلفن نیاز به استفاده کامل از TAPI نیست و می توان از قابلیتهای خود ویندوز در این زمینه استفاده کرد (Assisted Telephony) . در بخشهای بعدی در مورد سطوح مختلف سرویس در TAPI بیشتر صحبت خواهم کرد .
دومین مطلبی که باید مورد توجه قرار داد اینست که می خواهیم از TAPI 2.x استفاده کنیم یا از TAPI 3.x . تفاوت ایندو آنست که TAPI ورژن ۲ یک API برمبنای C است در حالیکه ورژن ۳ آن بر مبنای تکنولوژی COM می باشد . در بخشهای بعدی مطالب بیشتری در مورد تفاوتهای این دو نسخه بیان خواهم کرد .
بخشهای اصلی یک برنامه کامل TAPI عبارتند از :

۱ - TAPI Initialization : شامل load کردن TAPI dll ، اتصال به Server TAPI ، مذاکره در مورد ورژن TAPI و برپاسازی سیستم اطلاع رسانی event می باشد .

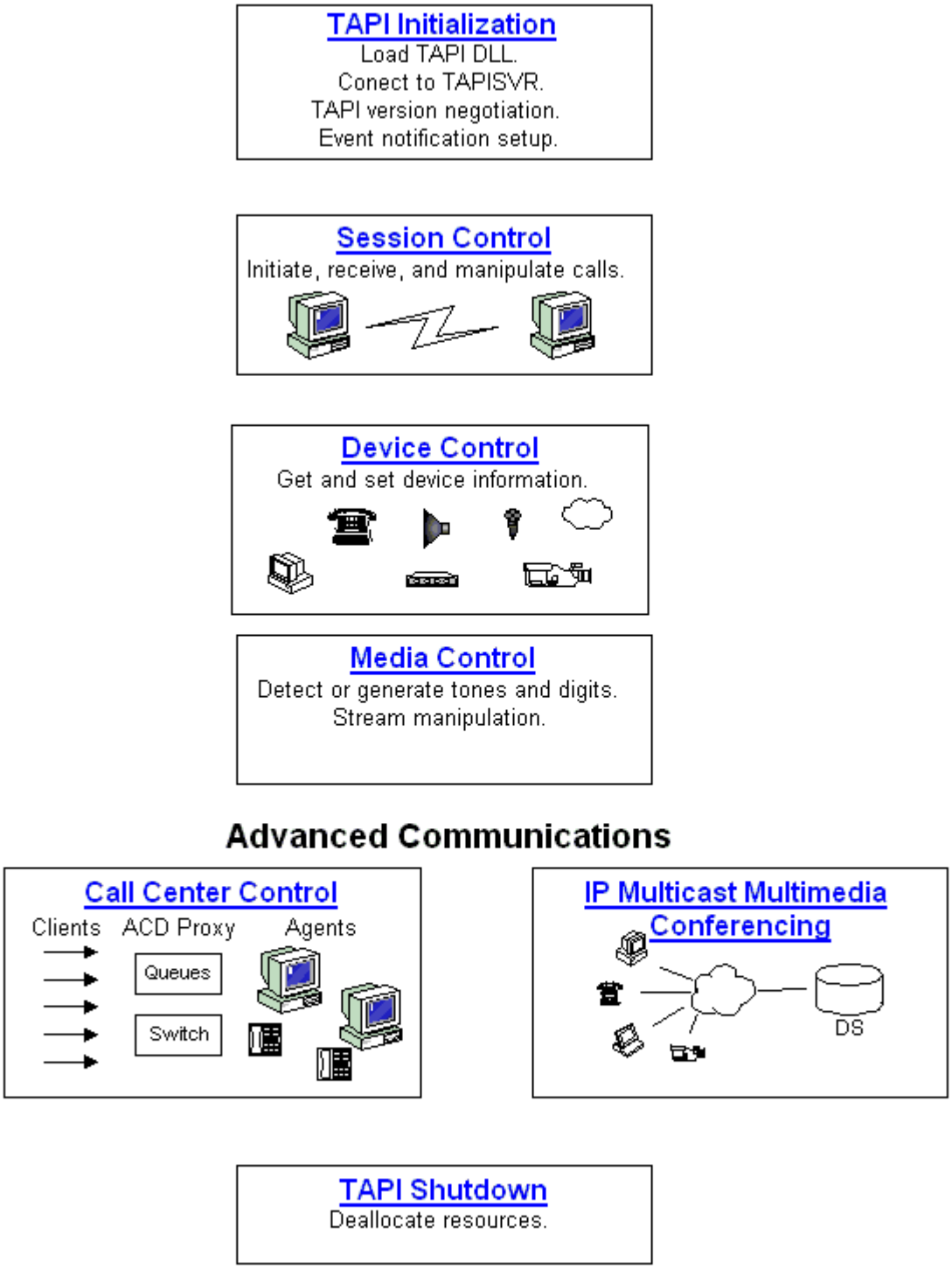
۲ - Session Control : مقداردهی اولیه ، دریافت و کنترل تماسها

۳ - Device Control : دریافت و تنظیم اطلاعات دستگاه

۴ - Media Control : تشخیص و یا تولید تونها و ارقام ، کنترل stream

۵ - TAPI Shutdown : آزاد سازی منابع

دیاگرام زیر این مراحل را بهتر نشان می دهد :



در بخشهای بعدی در مورد هر یک از این مراحل بیشتر صحبت خواهم کرد .

TAPI در ویژوال بیسیک - بخش دوم

مقداردهی اولیه TAPI :

عملکرد درست اجزای TAPI نیاز به برپاسازی محیط ارتباطی روی کامپیوتر مورد نظر دارد . مراحل این امر عبارتند از :

۱ - نصب TAPI : زمانیکه سخت افزار و یا نرم افزار برای اولین بار به کامپیوتر اضافه می شود انجام می گیرد . جزئیات کار به سیستم عامل و نرم افزار بستگی دارد .

۲ - مقداردهی ابتدائی : ساخت اشیا و مسیرهای ارتباطی

۳ - مذاکره در مورد ورژن TAPI : برای اطمینان از اینکه اجزای TAPI قادر به تبادل داده ها باشند .

۴ - استخراج اطلاعات منابع : بدست آوردن اطلاعاتی در مورد دستگاهی که می توان از آن در برنامه TAPI مورد نظرمان استفاده نمود .

۵ - Event notification : برپاسازی سیستم اطلاع رسانی event

TAPI در ویژوال بیسیک - بخش سوم

مقداردهی اولیه TAPI در ویژوال بیسیک :

از منوی Project گزینه References را انتخاب کرده و از لیست مربوطه مورد Microsoft TAPI 3.0 Type Library را انتخاب کنید .
حال وارد بخش کد نویسی فرمتان شوید و متغیر objTAPI را بصورت زیر تعریف کنید :

```
Dim objTapi As TAPI
```

سپس در بخش مربوط به Load Form شی objTAPI را بصورت زیر ایجاد می کنیم :

```
Set objTapi = New TAPI
```

همانطور که در بخشهای قبلی گفته شد ، قبل از فراخوانی هر تابع TAPI ابتدا بایستی آنرا مقداردهی اولیه کنیم . برای مقداردهی اولیه کردن شی TAPI عبارت زیر را بنویسید :

```
Call objTapi.Initialize
```

TAPI در ویژوال بیسیک - بخش چهارم

انتخاب یک آدرس :
کد زیر نشان می دهد که چگونه می توان با استفاده از شی TAPI در ویژوال بیسیک منابع تلفنی در دسترس را برای یک آدرس که بتواند یک مجموعه مشخص از نیازها را مدیریت کند ، بررسی کرد .
توجه داشته باشید که قبل از انجام این کار بایستی عمل مقداردهی اولیه TAPI را که در بخش قبل رررسی شد ، انجام دهید .

نکته : در کد زیر عمل error checking انجام نگرفته است و برای استفاده از کد زیر در برنامه های واقعی بایستی بخش بررسی خطا را به آن اضافه کنید .
۱ - تعریف یک شی آدرس و یک شی مجموعه آدرس :

```
Dim gobjAddress As ITAddress
As ITCollection Dim objCollAddresses
```

۲ - تنظیم شی objCollAddress بعنوان یک مجموعه آدرس از شی objTapi :

```
Set objCollAddresses = objTapi.Addresses
```

۳ - پیدا کردن آدرسی که بتواند از واسط مورد نظر ما پشتیبانی کند :

```
bFound = False
objCollAddresses.Count For indexAddr = 1 To
objCollAddresses.Item(indexAddr)x = Set objCrtAddress
Set objMediaSupport = objCrtAddress
objAddressCapabilities = objCrtAddress Set

nSelectedType ) x )If objMediaSupport.QueryMediaType
bFound = True
End If

Nothing = Set objAddressCapabilities
Set objMediaSupport = Nothing
Nothing = Set objCrtAddress

If bFound = True Then Exit For
Next indexAddr
```

در صورتیکه آدرس مورد نظر پیدا شود برنامه از حلقه خارج شده و gobjAddress یک آدرس قابل استفاده خواهد بود :

```
objcollAddresses.Item(indexAddr)x = Set gobjAddress
```

TAPI در ویژوال بیسیک - بخش پنجم

انجام Event Handling در TAPI :

کد زیر شامل یک event handler ساده برای TAPI ، رجیستر کردن واسط event ، تنظیم فیلتر event و رجیستر کردن تمام فراخوانیهای دادن اخطار است . هدف اصلی از این کد اینست که مطمئن شویم بخشی از TAPI که event ها را دریافت می کند پردازشی را قبل از انتقال به بخشهای دیگر انجام دهد .

تعاریفها :

```
TAPI Dim WithEvents gobjTapiWithEvents As
Attribute gobjTapiWithEvents.VB_VarHelpID = -1
glRegistrationToken As Long Dim



Const TAPI3_CALL_EVENTS =TE_CALLMEDIA Or
TE_CALLNOTIFICATION Or TE_CALLSTATE
```

تنظیم eventfilter بصورتیکه تمام event های تعریف شده برای TAPI را بپذیرد :

```
TAPI3_CALL_EVENTS = objTapi.EventFilter
```

رجیستر کردن event ها :

```
Set gobjTapiWithEvents = objTapi
Boolean, fMonitor As Boolean Dim fOwner As
Long Dim lMediaTypes As Long, lCallbackInstance As

fOwner = True
fOwner = True
fMonitor = False
TAPIMEDIATYPE_AUDIO = lMediaTypes
lCallbackInstance = 1

,gobjTapi.RegisterCallNotifications(gobjAddress,fMonitor = glRegistrationToken
fOwner,lMediaTypes,lCallbackInstance)x
```

TAPI در ویژوال بیسیک - بخش ششم

انتخاب یک ترمینال :

+ قبل از اینکه یک ترمینال را برای برقراری ارتباط انتخاب کنید بایستی TAPI Initialization و عمل انتخاب آدرس را انجام داده باشید .

ابتدا یک متغیر از نوع ITBasicCallControl (واسط کنترل تماس) تعریف می کنیم :

```
Dim objCallControl As ITBasicCallControl
objCallControl = gobjReceivedCallInfo Set
```

سپس یک متغیر از نوع ITTerminalSupport (کوئری از شی آدرس) تعریف می کنیم :

```
Dim objTerminalSupport As ITTerminalSupport
objTerminalSupport = gobjAddress Set
```

سپس متغیر ترمینال را تعریف کرده و توسط شی objTerminalSupport یک ترمینال را برای آن استخراج می کنیم :

```
Dim objTerminal As ITTerminal
objTerminalSupport.GetDefaultStaticTerminal(lMediaType, dir)x = Set objTerminal
```

در اینجا دیگر نیازی به شی objTerminalSupport نیست بنابراین آنرا آزاد می کنیم :

```
Set objTerminalSupport = Nothing
```

سپس نیاز به تعریف شی objStreamControl برای کنترل ترمینال است :

```
Dim objStreamControl As ITStreamControl
objStreamControl = objCallControl Set
```

در صورتیکه این شی ایجاد شود ، به ازای استریم های موجود در ITCollection امکان ایجاد ترمینال در یک حلقه for بررسی می شود و ترمینال مناسب انتخاب می گردد :

```
If Not (objStreamControl Is Nothing) Then
objITCollStreams As ITCollection Dim

objStreamControl.Streams = Set objITCollStreams
```

```
ITStream Dim nIndex As Long, objCrtStream As

For nIndex = 1 To objITCollStreams.Count
objITCollStreams.Item(nIndex)x = Set objCrtStream
Then If objCrtStream.MediaType = lMediaType
If objCrtStream.Direction = dir Then
objCrtStream.SelectTerminal(objTerminal)x Call
End If
End If
objCrtStream = Nothing Set
Next nIndex

Nothing = Set objITCollStreams
Set objStreamControl = Nothing
End If
```

ایجاد یک تماس (Make a Call) :

+ قبل از این بخش بایستی مراحل TAPI Initialization و عمل انتخاب آدرس انجام شده باشد .

این بخش برای ایجاد یک شی تماس ، بررسی و مشخص کردن استریمی که با این تماس در ارتباط است ، انتخاب و ایجاد ترمینالهای مناسب و کامل کردن ارتباط استفاده می شود .

قبل TAPI Initialization و عمل انتخاب آدرس و انتخاب ترمینال انجام شده باشد .

در ابتدا با استفاده از متد CreateCall یک شی تماس ساخته می شود :

```
nSelectedType,lMediaTypes)x ,gobjOrigAddress.CreateCall(strDestAddress = Set gobjCall
```

سپس در اینجا بایستی کدی که در بخش اول این درس برای انتخاب ترمینال نوشته شد آورده شود :

```
{
Select Terminal Code
}
```

سپس بایستی دستور Connect اجرا شود :

```
gobjCall.Connect (False)x
```

False بدین معناست که ارتباط بصورت آسنکرون برقرار می شود .

TAPI در ویژوال بیسیک - بخش پایانی

دریافت یک تماس :

کد زیر برای یافتن و یا ایجاد یک ترمینال مناسب برای دریافت یک تماس بکار می رود . بایستی توجه داشته باشید که قبل از اجرای کد زیر بایستی مراحل مقداردهی اولیه ، انتخاب یک آدرس و رجیسر کردن event ها را انجام دهید .
همچنین در کد زیر بایستی مرحله انتخاب ترمینال را نیز انجام دهید . توجه داشته باشید که در کد زیر متغیر pEvent یک اشاره گر برای واسط ITCallNotificationEvent است که توسط TAPI به event Handler داده می شود :

```
If TapiEvent = TE_CALLNOTIFICATION Then
objCallNotificationEvent As ITCallNotificationEvent Dim
objCallNotificationEvent = pEvent Set
ITCallInfo Dim gobjReceivedCallInfo As
Set gobjReceivedCallInfo = objCallNotificationEvent.Call
objCallControl As ITBasicCallControl Dim
gobjReceivedCallInfo = Set objCallControl
objCallControl.Answer
End If
```