```
 1  using System;
 2  using System.Runtime.InteropServices;
 3  using System.IO;
 4  using System.Collections.Generic;
 5  using System.Windows;
 6
 7  namespace SendFileTo
 8  {
 9      class MAPI
10      {
11          public bool AddRecipientTo(string email)
12          {
13              return AddRecipient(email, HowTo.MAPI_TO);
14          }
15
16          public bool AddRecipientCC(string email)
17          {
18              return AddRecipient(email, HowTo.MAPI_TO);
19          }
20
21          public bool AddRecipientBCC(string email)
22          {
23              return AddRecipient(email, HowTo.MAPI_TO);
24          }
25
26          public void AddAttachment(string strAttachmentFileName)
27          {
28              m_attachments.Add(strAttachmentFileName);
29          }
30
31          public int SendMailPopup(string strSubject, string strBody)
32          {
33              return SendMail(strSubject, strBody, MAPI_LOGON_UI | MAPI_DIALOG);
34          }
35
36          public int SendMailDirect(string strSubject, string strBody)
37          {
38              return SendMail(strSubject, strBody, MAPI_LOGON_UI);
39          }
40
41
42          [DllImport("MAPI32.DLL")]
43          static extern int MAPISendMail(IntPtr sess, IntPtr hwnd, MapiMessage
              message, int flg, int rsv);
44
45          int SendMail(string strSubject, string strBody, int how)
46          {
47              MapiMessage msg = new MapiMessage();
48              msg.subject = strSubject;
49              msg.noteText = strBody;
50
51              msg.recips = GetRecipients(out msg.recipCount);
52              msg.files = GetAttachments(out msg.fileCount);
53
54              m_lastError = MAPISendMail(new IntPtr(0), new IntPtr(0), msg, how,
                    0);
```

```
55              if (m_lastError > 1)
56                  MessageBox.Show("MAPISendMail failed! " + GetLastError(),
                      "MAPISendMail");
57
58              Cleanup(ref msg);
59              return m_lastError;
60          }
61
62          bool AddRecipient(string email, HowTo howTo)
63          {
64              MapiRecipDesc recipient = new MapiRecipDesc();
65
66              recipient.recipClass = (int)howTo;
67              recipient.name = email;
68              m_recipients.Add(recipient);
69
70              return true;
71          }
72
73          IntPtr GetRecipients(out int recipCount)
74          {
75              recipCount = 0;
76              if (m_recipients.Count == 0)
77                  return IntPtr.Zero;
78
79              int size = Marshal.SizeOf(typeof(MapiRecipDesc));
80              IntPtr intPtr = Marshal.AllocHGlobal(m_recipients.Count * size);
81
82              Int64 ptr = (Int64)intPtr;
83              foreach (MapiRecipDesc mapiDesc in m_recipients)
84              {
85                  Marshal.StructureToPtr(mapiDesc, (IntPtr)ptr, false);
86                  ptr += size;
87              }
88
89              recipCount = m_recipients.Count;
90              return intPtr;
91          }
92
93          IntPtr GetAttachments(out int fileCount)
94          {
95              fileCount = 0;
96              if (m_attachments == null)
97                  return IntPtr.Zero;
98
99              if ((m_attachments.Count <= 0) || (m_attachments.Count >
                    maxAttachments))
100                 return IntPtr.Zero;
101
102             int size = Marshal.SizeOf(typeof(MapiFileDesc));
103             IntPtr intPtr = Marshal.AllocHGlobal(m_attachments.Count * size);
104
105             MapiFileDesc mapiFileDesc = new MapiFileDesc();
106             mapiFileDesc.position = -1;
107             Int64 ptr = (Int64)intPtr;
108
```

```
109             foreach (string strAttachment in m_attachments)
110             {
111                 mapiFileDesc.name = Path.GetFileName(strAttachment);
112                 mapiFileDesc.path = strAttachment;
113                 Marshal.StructureToPtr(mapiFileDesc, (IntPtr)ptr, false);
114                 ptr += size;
115             }
116 
117             fileCount = m_attachments.Count;
118             return intPtr;
119         }
120 
121         void Cleanup(ref MapiMessage msg)
122         {
123             int size = Marshal.SizeOf(typeof(MapiRecipDesc));
124             Int64 ptr = 0;
125 
126             if (msg.recips != IntPtr.Zero)
127             {
128                 ptr = (Int64)msg.recips;
129                 for (int i = 0; i < msg.recipCount; i++)
130                 {
131                     Marshal.DestroyStructure((IntPtr)ptr, typeof
                    (MapiRecipDesc));
132                     ptr += size;
133                 }
134                 Marshal.FreeHGlobal(msg.recips);
135             }
136 
137             if (msg.files != IntPtr.Zero)
138             {
139                 size = Marshal.SizeOf(typeof(MapiFileDesc));
140 
141                 ptr = (Int64)msg.files;
142                 for (int i = 0; i < msg.fileCount; i++)
143                 {
144                     Marshal.DestroyStructure((IntPtr)ptr, typeof
                    (MapiFileDesc));
145                     ptr += size;
146                 }
147                 Marshal.FreeHGlobal(msg.files);
148             }
149 
150             m_recipients.Clear();
151             m_attachments.Clear();
152             m_lastError = 0;
153         }
154 
155         public string GetLastError()
156         {
157             if (m_lastError <= 26)
158                 return errors[ m_lastError ];
159             return "MAPI error [" + m_lastError.ToString() + "]";
160         }
161 
162         readonly string[] errors = new string[] {
```

```
163         "OK [0]", "User abort [1]", "General MAPI failure [2]", "MAPI login
              failure [3]",
164         "Disk full [4]", "Insufficient memory [5]", "Access denied [6]", "-
              unknown- [7]",
165         "Too many sessions [8]", "Too many files were specified [9]", "Too
              many recipients were specified [10]", "A specified attachment was
              not found [11]",
166         "Attachment open failure [12]", "Attachment write failure [13]",
              "Unknown recipient [14]", "Bad recipient type [15]",
167         "No messages [16]", "Invalid message [17]", "Text too large [18]",
              "Invalid session [19]",
168         "Type not supported [20]", "A recipient was specified ambiguously
              [21]", "Message in use [22]", "Network failure [23]",
169         "Invalid edit fields [24]", "Invalid recipients [25]", "Not supported
              [26]"
170         };
171
172
173         List<MapiRecipDesc> m_recipients    = new List<MapiRecipDesc>();
174         List<string> m_attachments  = new List<string>();
175         int m_lastError = 0;
176
177         const int MAPI_LOGON_UI = 0x00000001;
178         const int MAPI_DIALOG = 0x00000008;
179         const int maxAttachments = 20;
180
181         enum HowTo{MAPI_ORIG=0, MAPI_TO, MAPI_CC, MAPI_BCC};
182     }
183
184     [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi)]
185     public class MapiMessage
186     {
187         public int reserved;
188         public string subject;
189         public string noteText;
190         public string messageType;
191         public string dateReceived;
192         public string conversationID;
193         public int flags;
194         public IntPtr originator;
195         public int recipCount;
196         public IntPtr recips;
197         public int fileCount;
198         public IntPtr files;
199     }
200
201     [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi)]
202     public class MapiFileDesc
203     {
204         public int reserved;
205         public int flags;
206         public int position;
207         public string path;
208         public string name;
209         public IntPtr type;
210     }
```

```
211
212     [StructLayout(LayoutKind.Sequential, CharSet=CharSet.Ansi)]
213     public class MapiRecipDesc
214     {
215         public int      reserved;
216         public int      recipClass;
217         public string   name;
218         public string   address;
219         public int      eIDSize;
220         public IntPtr   entryID;
221     }
222 }
223
```