```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++

$Id: udp.c,v 1.16 2001/09/02 04:49:03 davidc Exp $

Copyright (c) 2001 BeComm Corporation

Filename:

    udp.c

Abstract:

    The implementation of the UDP portion of the SocketLib API. This is
the
    Win32 (NT/95/CE) version using the standard sockets interface exported
    by WinSock.

Owner:

    David Costanzo (davidc)

------------------------------------------------------------------------------
---*/

#include <windows.h>
#include <winsock2.h>
#include <process.h>
#include <assert.h>

#define SOS_DEBUG_ZONE "/classes/socket/udp"
#include <sosstrings.h>
#include <sosisocket.h>
#include "socket.h"
#include "udp.h"

SOS_SOURCE_VERSION("$Id: udp.c,v 1.16 2001/09/02 04:49:03 davidc Exp $");

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
macros
------------------------------------------------------------------------------
---*/

#define ERROR_BIND_FAILED           SOS_Error
```

```
41 #define ERROR_SOCKET_FAILED         SOS_Error
42 #define ERROR_BIND_FAILED           SOS_Error
43 #define ERROR_CONNECT_FAILED        SOS_Error
44 #define ERROR_GETSOCKNAME_FAILED    SOS_Error
45 #define ERROR_SETSOCKOPT_FAILED     SOS_Error
46
47 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++++
48 types
49 ------------------------------------------------------------------------------
… ---*/
50
51 typedef enum _UDP_CONNECTION_FUNCTION_ID {
52     UDP_CONNECTION_FUNCTION_ID_OnReceive,
53     UDP_CONNECTION_FUNCTION_ID_OnClose,
54 } UDP_CONNECTION_FUNCTION_ID;
55
56 typedef struct _UDP_CONNECTION_PARAM UDP_CONNECTION_PARAM;
57
58
59 struct _UDP_CONNECTION_PARAM {
60
61     UDP_CONNECTION_FUNCTION_ID  FunctionId;
62
63     union _UDP_CONNECTION_PARAMS {
64
65         struct _UDP_CONNECTION_PARAM_ON_RECEIVE {
66             SOS_ISOCKET_UDP *           Socket;
67             const SOS_ISOCKET_ADDRESS*  SocketAddress;
68             void*                       Buffer;
69             size_t                      BufferLength;
70             SOS_FREEPROC                BufferFree;
71         } OnReceive;
72
73         struct _UDP_CONNECTION_PARAM_ON_CLOSE {
74             SOS_ISOCKET_UDP *           Socket;
75         } OnClose;
76
77     } Params;
78 };
79
80
81 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++++
Constants
```

```
83  ------------------------------------------------------------------------
…   ---*/
84  /* REVISIT: it would be nice if we could check the size
85     of the message and allocate the right amount for each
86     message instead of having to keep a big buffer around
87     and copy to a buffer of the right size each time... */
88  static const size_t g_UdpRecvSize = 65535;
89
90
91  /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…   +++++
92  Data Types
93  ------------------------------------------------------------------------
…   ---*/
94
95  /* UDP socket context to be passed back and forth with caller of this
…   library */
96  struct _SOS_ISOCKET_UDP {
97      SOCKET                          NativeSocket;
98      int                             RefCount;
99      SOS_ISOCKET_UDP_RECEIVE         RecvCallback;
100     SOS_ISOCKET_CLOSED              ClosedCallback;
101     struct sockaddr_in              RemoteSockAddr;
102     SOS_ISOCKET_ADDRESS             SocketAddr;
103     void*                           UserContext;
104     SOS_BOOLEAN                     IsCopy;
105
106     unsigned long                   NativeRecvThread;
107 };
108
109 struct _SOS_ISOCKET_UDP_LISTEN {
110     SOS_ISOCKET_UDP             Socket;
111 };
112
113
114
115 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…   +++++
116 Internal Routines
117 ------------------------------------------------------------------------
…   ---*/
118
119 /*++
120 Routine Name:
```

```
122     UdpSocketCreate
123
124 Routine Description:
125
126     This routine allocates and initializes an SOS_ISOCKET_UDP struct.
127     It does not call any sockets routines.
128     The structure is returned with a reference count of 1.
129
130 Parameters:
131
132     SOCKET NativeSocket - [in]
133         The native socket handle.
134
135     SOS_ISOCKET_UDP_RECEIVE RecvCallback - [in]
136         The user-defined RecvCallback.
137
138     SOS_ISOCKET_CLOSED ClosedCallback - [in]
139         The user-defined ClosedCallback.
140
141     const struct sockaddr_in* RemoteSockAddr - [in]
142         The remote socket address.
143         This parameter is optional.
144
145     const SOS_ISOCKET_ADDRESS* SocketAddr - [in]
146         The socket address.
147
148     void* UserContext - [in]
149         The user context to pass into RecvCallback and ClosedCallback.
150
151 Return Value:
152
153     SOS_ISOCKET_UDP* -
154         A pointer to a newly allocated SOS_ISOCKET_UDP structure with
155         a reference count of 1.
156         NULL, if the structure could not be created.
157
158 --*/
159 static
160 SOS_ISOCKET_UDP*
161 UdpSocketCreate(
162     SOCKET                      NativeSocket,
163     SOS_ISOCKET_UDP_RECEIVE     RecvCallback,
164     SOS_ISOCKET_CLOSED          ClosedCallback,
165     const struct sockaddr_in *  RemoteSockAddr,
```

```
167     void*                       UserContext
168 )
169 {
170     SOS_ISOCKET_UDP*  udpSocket;
171 
172     /* allocate enough size for the larger SOS_ISOCKET_UDP_LISTEN */
173     udpSocket = malloc(sizeof(SOS_ISOCKET_UDP_LISTEN));
174     if (udpSocket != NULL) {
175 
176         udpSocket->NativeSocket   = NativeSocket;
177         udpSocket->ClosedCallback = ClosedCallback;
178         udpSocket->RecvCallback   = RecvCallback;
179         if (RemoteSockAddr) {
180             udpSocket->RemoteSockAddr = *RemoteSockAddr;
181         }
182         udpSocket->SocketAddr     = *SocketAddr;
183         udpSocket->UserContext    = UserContext;
184         udpSocket->IsCopy         = SOS_False;
185 
186         udpSocket->RefCount       = 1;
187     }
188 
189     return udpSocket;
190 }
191 
192 /*++
193 Routine Name:
194 
195     UdpSocketDestroy
196 
197 Routine Description:
198 
199     This is the SOS_FREEPROC for udp sockets.  It simply unallocated
200     the structure.  It does not make any socket-specific calls, such
201     as closing the socket handle.
202 
203 Parameters:
204 
205     SOS_ISOCKET_UDP* UdpSocket - [consumed]
206         The UdpSocket structure to destroy.
207 
208 Return Value:
209 
210     None
```

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.