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

$Id: udp.c,v 1.10 2001/09/02 05:21:54 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 <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <pthread.h>
#include <assert.h>
#include <errno.h>

#include <sys/time.h>
#include <unistd.h>

#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>

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

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

```
43 SOS_SOURCE_VERSION("$Id: udp.c,v 1.10 2001/09/02 05:21:54 davidc Exp $");
44
45 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…  +++++
46 macros
47 ----------------------------------------------------------------------------
…  ---*/
48
49 #define ERROR_BIND_FAILED           SOS_Error
50 #define ERROR_SOCKET_FAILED         SOS_Error
51 #define ERROR_BIND_FAILED           SOS_Error
52 #define ERROR_CONNECT_FAILED        SOS_Error
53 #define ERROR_GETSOCKNAME_FAILED    SOS_Error
54 #define ERROR_SETSOCKOPT_FAILED     SOS_Error
55
56 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…  +++++
57 types
58 ----------------------------------------------------------------------------
…  ---*/
59
60 typedef enum _UDP_CONNECTION_FUNCTION_ID {
61     UDP_CONNECTION_FUNCTION_ID_OnReceive,
62     UDP_CONNECTION_FUNCTION_ID_OnClose,
63 } UDP_CONNECTION_FUNCTION_ID;
64
65 typedef struct _UDP_CONNECTION_PARAM UDP_CONNECTION_PARAM;
66
67
68 struct _UDP_CONNECTION_PARAM {
69
70     UDP_CONNECTION_FUNCTION_ID  FunctionId;
71
72     union _UDP_CONNECTION_PARAMS {
73
74         struct _UDP_CONNECTION_PARAM_ON_RECEIVE {
75             SOS_ISOCKET_UDP *           Socket;
76             const SOS_ISOCKET_ADDRESS * SocketAddress;
77             void*                       Buffer;
78             size_t                      BufferLength;
79             SOS_FREEPROC                BufferFree;
80         } OnReceive;
81
82         struct _UDP_CONNECTION_PARAM_ON_CLOSE {
```

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

```
84          } OnClose;
85
86      } Params;
87  };
88
89  typedef int SOCKET;
90
91  /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…   +++++
92  Constants
93  ----------------------------------------------------------------------
…   ---*/
94  /* REVISIT: it would be nice if we could check the size
95     of the message and allocate the right amount for each
96     message instead of having to keep a big buffer around
97     and copy to a buffer of the right size each time... */
98  static const size_t g_UdpRecvSize = 65535;
99
100
101 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…   +++++
102 Data Types
103 ----------------------------------------------------------------------
…   ---*/
104
105 /* UDP socket context to be passed back and forth with caller of this
…   library */
106 struct _SOS_ISOCKET_UDP {
107     SOCKET                      NativeSocket;
108
109     pthread_mutex_t             Mutex;          /* non-recursive */
110     int                         RefCount;
111
112     SOS_ISOCKET_UDP_RECEIVE     RecvCallback;
113     SOS_ISOCKET_CLOSED          ClosedCallback;
114     struct sockaddr_in          RemoteSockAddr;
115     SOS_ISOCKET_ADDRESS         SocketAddr;
116     void*                       UserContext;
117     SOS_BOOLEAN                 IsCopy;
118
119     pthread_t                   NativeRecvThread;
120 };
121
122 struct _SOS_ISOCKET_UDP_LISTEN {
```

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

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

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

```
167         a reference count of 1.
168         NULL, if the structure could not be created.
169 
170 --*/
171 static
172 SOS_ISOCKET_UDP*
173 UdpSocketCreate(
174     SOCKET                      NativeSocket,
175     SOS_ISOCKET_UDP_RECEIVE     RecvCallback,
176     SOS_ISOCKET_CLOSED          ClosedCallback,
177     const struct sockaddr_in *  RemoteSockAddr,
178     const SOS_ISOCKET_ADDRESS * SocketAddr,
179     void*                       UserContext
180 )
181 {
182     SOS_ISOCKET_UDP*  udpSocket;
183 
184     /* allocate enough size for the larger SOS_ISOCKET_UDP_LISTEN */
185     udpSocket = malloc(sizeof(SOS_ISOCKET_UDP_LISTEN));
186     if (udpSocket != NULL) {
187 
188         int initStatus;
189 
190         initStatus = pthread_mutex_init(&udpSocket->Mutex, NULL);
191         if (initStatus == 0) {
192 
193             udpSocket->NativeSocket   = NativeSocket;
194             udpSocket->ClosedCallback = ClosedCallback;
195             udpSocket->RecvCallback   = RecvCallback;
196             if (RemoteSockAddr) {
197                 udpSocket->RemoteSockAddr = *RemoteSockAddr;
198             }
199             udpSocket->SocketAddr     = *SocketAddr;
200             udpSocket->UserContext    = UserContext;
201             udpSocket->IsCopy         = SOS_False;
202 
203             udpSocket->RefCount       = 1;
204         }
205         else {
206             free(udpSocket);
207             udpSocket = NULL;
208         }
209     }
210 
211     return udpSocket;
```

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

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.