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

Copyright (c) 2001 BeComm Corporation

Filename:

    fanout.c

Group Name:

    todo

Group Overview:

    todo

Owner:

    Guy Carpenter (guyc) 28-Sep-2001

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

#define SOS_DEBUG_ZONE "/beads/fanout"

#include <sosstrings.h>
#include <sosmultimedia.h>

SOS_SOURCE_VERSION (
    "$Id: fanout.c,v 1.2 2001/10/22 23:31:42 guyc Exp $"
);

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
Named Constants
--------------------------------------------------------------------------------
---*/

/*
 * Name of bead
 */
static const char BEAD_NAME[] = "fanout";

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

```
41… +++++
42  Structs
43  -------------------------------------------------------------------------
…   ---*/
44
45  typedef struct {
46      SOS_UINT32              PathCount;
47      SOS_PATH **             PathList;
48  } FANOUT_CONTEXT;
49
50  /*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…   +++++
51  Context Stuff
52  -------------------------------------------------------------------------
…   ---*/
53
54  static
55  void
56  FanOut_ContextDestroy(
57      FANOUT_CONTEXT *        Context
58  )
59  {
60      SOS_DEBUGOUT_FUNC_TRACE("FanOut_ContextDestroy\n");
61
62      if (Context) {
63          if (Context->PathList) {
64              SOS_UINT32 i;
65              for (i=0;i<Context->PathCount;i++) {
66                  SOS_Path_Destroy(Context->PathList[i]);
67              }
68              SOS_Mem_Free(Context->PathList);
69          }
70          SOS_Mem_Free(Context);
71      }
72  }
73
74  static
75  FANOUT_CONTEXT *
76  FanOut_ContextCreate(
77      void
78  )
79  {
80      FANOUT_CONTEXT *context;
81
82      SOS_DEBUGOUT_FUNC_TRACE("FanOut_ContextCreate\n");
```

```
83
84      context = SOS_Mem_Alloc(sizeof(*context));
85      if (context) {
86          SOS_memset(context, 0, sizeof(*context));
87      }
88
89      return context;
90  }
91
92  static
93  SOS_STATUS
94  FanOut_ContextInit(
95      FANOUT_CONTEXT *     Context,
96      SOS_PATH *           ParentPath,
97      SOS_UINT32           PathCount
98  )
99  {
100     SOS_STATUS status = SOS_Success;
101     SOS_UINT32 i;
102
103     Context->PathCount = PathCount;
104     Context->PathList = SOS_Mem_Alloc(sizeof(SOS_PATH*)*PathCount);
105
106     if (Context->PathList) {
107         SOS_memset(Context->PathList, 0, sizeof(SOS_PATH*)*PathCount);
108         for (i=0;i<PathCount && SOS_SUCCEEDED(status);i++) {
109             Context->PathList[i]=SOS_Path_Create(ParentPath);
110             if (Context->PathList[i]) {
111                 SOS_REGOBJECT *object = SOS_UInt32_Create(i);
112                 SOS_Path_AttributeSet(
113                     Context->PathList[i],
114                     "FanoutIndex",
115                     object
116                 );
117                 SOS_RegObject_Release(object);
118             } else {
119                 status = SOS_Error;
120             }
121         }
122     } else {
123         /* no pathlist */
124         status = SOS_Error;
125     }
126
127     return status;
```

```
128 }
129
130 static
131 SOS_STATUS
132 FanOut_MessageSend(
133     FANOUT_CONTEXT *      Context,
134     SOS_MESSAGE *         Message
135 )
136 {
137     SOS_STATUS status = SOS_Success;
138
139     if (Context->PathList) {
140         SOS_UINT32 i;
141         for (i=0;i<Context->PathCount && SOS_SUCCEEDED(status);i++) {
142             SOS_MESSAGE *copy;
143             SOS_Message_HeadMessageCopyFrom(Message,SOS_UINT32_MAX,
…&copy);
144             status = SOS_Path_MessageSend(Context->PathList[i], copy);
145         }
146     } else {
147         status = SOS_Error;
148     }
149     return status;
150 }
151
152 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…+++++
153
154 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…+++*/
155
156 static
157 SOS_STATUS
158 FanOut_KeyCreate(
159     SOS_PATH     *Path,
160     SOS_MESSAGE  *Message
161 )
162 {
163     SOS_STATUS status = SOS_Success;
164     static SOS_UINT32 s_UniqueId = 0;
165     SOS_REGOBJECT*  uniqueSessionKey;
166
167     SOS_DEBUGOUT_FUNC_TRACE("FanOut_KeyCreate\n");
168
```

```
170     SOS_Path_SessionKeySet(Path, uniqueSessionKey);
171     SOS_RegObject_Release(uniqueSessionKey);
172 
173     return status;
174 }
175 
176 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…   +++++
177 Bead Definition
178 ----------------------------------------------------------------------
…   ---*/
179 
180 static
181 SOS_STATUS
182 FanOut_MessageHandler(
183     SOS_PATH    *Path,
184     SOS_MESSAGE *Message
185 )
186 {
187     SOS_STATUS status = SOS_Success;
188     FANOUT_CONTEXT *context;
189 
190     SOS_DEBUGOUT_FUNC_TRACE("FanOut_MessageHandler\n");
191 
192     SOS_Path_SessionContextPeek(Path, (void**)&context);
193 
194     status = FanOut_MessageSend(context, Message);
195 
196     SOS_Message_Destroy(Message);
197 
198     return status;
199 }
200 
201 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…   +++++
202 Session Setup
203 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…   +++*/
204 
205 static
206 SOS_STATUS
207 FanOut_SessionInit(
208         SOS_PATH  *Path
209 )
```

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.