```
**********************************************************
    5928 Sun Jan 19 01:29:09 2020
new/usr/src/man/man3pool/pool_get_binding.3pool
12202 noise in example code in some section 3pool man pages
**********************************************************
```

```
   1 '\" te
   2 .\" Copyright (c) 2007, Sun Microsystems, Inc. All Rights Reserved.
   3 .\" The contents of this file are subject to the terms of the Common Development
   4 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
   5 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
   6 .TH POOL_GET_BINDING 3POOL "January 18, 2020"
   6 .TH POOL_GET_BINDING 3POOL "Mar 27, 2007"
   7 .SH NAME
   8 pool_get_binding, pool_set_binding, pool_get_resource_binding \- set and query
   9 process to resource pool bindings
  10 .SH SYNOPSIS
  11 .LP
  11 .nf
  12 cc [ \fIflag\fR\&.\|.\|. ] \fIfile\fR\&.\|.\|. \fB-lpool\fR [ \fIlibrary\fR\&.\|
  13 #include <pool.h>

  15 \fBchar *\fR\fBpool_get_binding\fR(\fBpid_t\fR \fIpid\fR);
  16 .fi

  18 .LP
  19 .nf
  20 \fBint\fR \fBpool_set_binding\fR(\fBconst char *\fR\fIpool\fR, \fBidtype_t\fR \f
  21      \fBid_t\fR \fIid\fR);
  22 .fi

  24 .LP
  25 .nf
  26 \fBchar *\fR\fBpool_get_resource_binding\fR(\fBconst char *\fR\fItype\fR, \fBpid
  27 .fi

  29 .SH DESCRIPTION
  31 .sp
  32 .LP
  30 The \fBpool_get_binding()\fR function returns the name of the pool on the
  31 running system that contains the set of resources to which the given process is
  32 bound. If no such pool exists on the system or the search returns more than one
  33 pool (since the set of resources is referred to by more than one pool),
  34 \fINULL\fR is  returned and the pool error value is set to
  35 \fBPOE_INVALID_SEARCH\fR.
  36 .sp
  37 .LP
  38 It is possible that one of the resources to which the given process is bound is
  39 not associated with a pool. This could occur if a processor set was created
  40 with one of the \fBpset_()\fR functions and the process was then bound to that
  41 set. It could also occur if the process was bound to a resource set not
  42 currently associated with a pool, since resources can exist that are not
  43 associated with a pool.
  44 .sp
  45 .LP
  46 The \fBpool_set_binding()\fR function binds the processes matching \fIidtype\fR
  47 and \fIid\fR to the resources associated with \fIpool\fR on the running system.
  48 This function requires the privilege required by the underlying resource types
  49 referenced by the pool; generally, this requirement is equivalent to requiring
  50 superuser privilege.
  51 .sp
  52 .LP
  53 The \fIidtype\fR parameter can be one of the following types:
  56 The \fIidtype\fR parameter can be of the following types:
  54 .sp
  55 .ne 2
  56 .na
```



```
  57 \fB\fBP_PID\fR\fR
  58 .ad
  59 .RS 12n
  60 The \fIid\fR parameter is a pid.
  61 .RE

  63 .sp
  64 .ne 2
  65 .na
  66 \fB\fBP_TASKID\fR\fR
  67 .ad
  68 .RS 12n
  69 The \fIid\fR parameter is a taskid.
  70 .RE

  72 .sp
  73 .ne 2
  74 .na
  75 \fB\fBP_PROJID\fR\fR
  76 .ad
  77 .RS 12n
  78 The \fIid\fR parameter is a project ID. All currently running processes
  79 belonging to the given project will be bound to the pool's resources.
  80 .RE

  82 .sp
  83 .LP
  84 The \fBpool_get_resource_binding()\fR function returns the name of the resource
  85 of the supplied type to which the supplied process is bound.
  86 .sp
  87 .LP
  88 The application must explicitly free the memory allocated for the return values
  91 The application must explicity free the memory allocated for the return values
  89 for \fBpool_get_binding()\fR and \fBpool_get_resource_binding()\fR.
  90 .SH RETURN VALUES
  94 .sp
  95 .LP
  91 Upon successful completion, \fBpool_get_binding()\fR returns the name of the
  92 pool to which the process is bound. Otherwise it returns \fINULL\fR and
  93 \fBpool_error\fR(3POOL) returns the pool-specific error value.
  94 .sp
  95 .LP
  96 Upon successful completion, \fBpool_set_binding()\fR returns \fBPO_SUCCESS\fR.
  97 Otherwise, it returns \fBPO_FAIL\fR and \fBpool_error()\fR returns the
  98 pool-specific error value.
  99 .sp
 100 .LP
 101 Upon successful completion, \fBpool_get_resource_binding()\fR returns the name
 102 of the resource of the specified type to which the process is bound. Otherwise
 103 it returns \fINULL\fR and \fBpool_error()\fR returns the pool-specific error
 104 value.
 105 .SH ERRORS
 111 .sp
 112 .LP
 106 The \fBpool_get_binding()\fR function will fail if:
 107 .sp
 108 .ne 2
 109 .na
 110 \fB\fBPOE_INVALID_CONF\fR\fR
 111 .ad
 112 .RS 22n
 113 The configuration is invalid.
 114 .RE

 116 .sp
 117 .ne 2
```

```
.na
\fB\fBPOE_INVALID_SEARCH\fR\fR
.ad
.RS 22n
It is not possible to determine the binding for this target due to the
overlapping nature of the pools configured for this system, or the pool could
not be located.
.RE

.sp
.ne 2
.na
\fB\fBPOE_SYSTEM\fR\fR
.ad
.RS 22n
A system error has occurred. Check the system error code for more details.
.RE

.sp
.LP
The \fBpool_set_binding()\fR function will fail if:
.sp
.ne 2
.na
\fB\fBPOE_INVALID_SEARCH\fR\fR
.ad
.RS 22n
The pool could not be found.
.RE

.sp
.ne 2
.na
\fB\fBPOE_INVALID_CONF\fR\fR
.ad
.RS 22n
The configuration is invalid.
.RE

.sp
.ne 2
.na
\fB\fBPOE_SYSTEM\fR\fR
.ad
.RS 22n
A system error has occurred. Check the system error code for more details.
.RE

.sp
.LP
The \fBpool_get_resource_binding()\fR function will fail if:
.sp
.ne 2
.na
\fB\fBPOE_INVALID_CONF\fR\fR
.ad
.RS 22n
The configuration is invalid.
.RE

.sp
.ne 2
.na
\fB\fBPOE_INVALID_SEARCH\fR\fR
.ad
.RS 22n
```



```
The target is not bound to a resource of the specified type.
.RE

.sp
.ne 2
.na
\fB\fBPOE_SYSTEM\fR\fR
.ad
.RS 22n
A system error has occurred. Check the system error code for more details.
.RE

.SH EXAMPLES
.LP
\fBExample 1 \fRBind the current process to the pool named "target".
.sp
.in +2
.nf
#include <sys/types.h>
#include <pool.h>
#include <unistd.h>

\&...

id_t pid = getpid();

\&...

if (pool_set_binding("target", P_PID, pid) == PO_FAIL) \{
        (void) fprintf(stderr, "pool binding failed (%d)\\n",
        (void) fprintf(stderr, "pool binding failed (\\%d)\\B{}n",
                pool_error());
\}
.fi
.in -2

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
CSI     Enabled
_
Interface Stability     Unstable
_
MT-Level        Safe
.TE

.SH SEE ALSO
.sp
.LP
\fBlibpool\fR(3LIB), \fBpool_error\fR(3POOL), \fBattributes\fR(5)
```

```
**********************************************************
    5423 Sun Jan 19 01:29:09 2020
new/usr/src/man/man3pool/pool_get_pool.3pool
12202 noise in example code in some section 3pool man pages
**********************************************************
```

```
1  '\" te
2  .\" Copyright (c) 2005, Sun Microsystems, Inc. All Rights Reserved.
3  .\" The contents of this file are subject to the terms of the Common Development
4  .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
5  .\" When distributing Covered Code, include this CDDL HEADER in each file and in
6  .TH POOL_GET_POOL 3POOL "January 18, 2020"
6  .TH POOL_GET_POOL 3POOL "Jul 18, 2005"
7  .SH NAME
8  pool_get_pool, pool_get_resource, pool_query_components, pool_query_pools,
9  pool_query_resources \- retrieve resource pool configuration elements
10 .SH SYNOPSIS
11 .LP
11 .nf
12 cc [ \fIflag\fR]\&.\|.\|.  \fIfile\fR\&.\|.\|. \fB-lpool\fR [ \fIlibrary\fR\&.\|
13 #include <pool.h>

15 \fBpool_t *\fR\fBpool_get_pool\fR(\fBpool_conf_t *\fR\fIconf\fR, \fBconst char *
16 .fi

18 .LP
19 .nf
20 \fBpool_resource_t *\fR\fBpool_get_resource\fR(\fBpool_conf_t *\fR\fIconf\fR
21      \fBconst char *\fR\fItype\fR, \fBconst char *\fR\fIname\fR);
22 .fi

24 .LP
25 .nf
26 \fBpool_component_t **\fR\fBpool_query_components\fR(\fBpool_conf_t *\fR\fIconf\
27      \fBuint_t *\fR\fInelem\fR, \fBpool_value_t **\fR\fIprops\fR);
28 .fi

30 .LP
31 .nf
32 \fBpool_t **\fR\fBpool_query_pools\fR(\fBpool_conf_t *\fR\fIconf\fR, \fBuint_t *
33      \fBpool_value_t **\fR\fIprops\fR);
34 .fi

36 .LP
37 .nf
38 \fBpool_component_t **\fR\fBpool_query_resources\fR(\fBpool_conf_t *\fR\fIconf\f
39      \fBuint_t *\fR\fInelem\fR, \fBpool_value_t **\fR\fIprops\fR);
40 .fi

42 .SH DESCRIPTION
44 .sp
45 .LP
43 These functions provide a means for querying the contents of the specified
44 configuration. The \fIconf\fR argument for each function refers to the target
45 configuration to which the operation applies.
46 .sp
47 .LP
48 The \fBpool_get_pool()\fR function returns the pool with the given name from
49 the provided configuration.
50 .sp
51 .LP
52 The \fBpool_get_resource()\fR function returns the resource with the given name
53 and type from the provided configuration.
54 .sp
55 .LP
56 The \fBpool_query_components()\fR function retrieves all resource components
57 that match the given list of properties. If the list of properties is
```



```
58 \fINULL\fR, all components are returned. The number of elements returned is
59 stored in the location pointed to by \fInelem\fR. The value returned by
60 \fBpool_query_components()\fR is allocated with \fBmalloc\fR(3C) and must be
61 explicitly freed.
62 .sp
63 .LP
64 The \fBpool_query_pools()\fR function behaves similarly to
65 \fBpool_query_components()\fR and returns the list of pools that match the
66 given list of properties. The value returned must be freed by the caller.
67 .sp
68 .LP
69 The \fBpool_query_resources()\fR function similarly returns the list of
70 resources that match the given list of properties. The return value must be
71 freed by the caller.
72 .SH RETURN VALUES
76 .sp
77 .LP
73 The \fBpool_get_pool()\fR and \fBpool_get_resource()\fR functions return the
74 matching pool and resource, respectively. Otherwise, they return \fINULL\fR and
75 \fBpool_error\fR(3POOL) returns the pool-specific error value.
76 .sp
77 .LP
78 The \fBpool_query_components()\fR, \fBpool_query_pools()\fR, and
79 \fBpool_query_resources()\fR functions return a null-terminated array of
80 components, pools, and resources, respectively. If the query was unsuccessful
81 or there were no matches, \fINULL\fR is returned and \fBpool_error()\fR returns
82 the pool-specific error value.
83 .SH ERRORS
84 The \fBpool_get_pool()\fR function will fail if:
85 .sp
90 .LP
91 The \fBpool_get_pool()\fR will fail if:
92 .sp
86 .ne 2
87 .na
88 \fB\fBPOE_BADPARAM\fR\fR
89 .ad
90 .RS 16n
91 The supplied configuration's status is not \fBPOF_VALID\fR.
92 .RE

94 .sp
95 .LP
96 The \fBpool_get_resource()\fR function will fail if:
103 The \fBpool_get_resource()\fR will fail if:
97 .sp
98 .ne 2
99 .na
100 \fB\fBPOE_BADPARAM\fR\fR
101 .ad
102 .RS 16n
103 The supplied configuration's status is not \fBPOF_VALID\fR.
104 .RE

106 .sp
107 .ne 2
108 .na
109 \fB\fBPOE_SYSTEM\fR\fR
110 .ad
111 .RS 16n
112 There is not enough memory available to allocate working buffers. Check
113 \fBerrno\fR for the specific system error code.
114 .RE

116 .sp
117 .LP
```

```
118 The \fBpool_query_components()\fR, \fBpool_query_pools()\fR, and
119 \fBpool_query_resources()\fR functions will fail if:
126 \fIpool_query_resources()\fR will fail if:
120 .sp
121 .ne 2
122 .na
123 \fB\fBPOE_BADPARAM\fR\fR
124 .ad
125 .RS 20n
126 The supplied configuration's status is not \fBPOF_VALID\fR.
127 .RE

129 .sp
130 .ne 2
131 .na
132 \fB\fBPOE_INVALID_CONF\fR\fR
133 .ad
134 .RS 20n
135 The query generated results that were not of the correct type. The
136 configuration is invalid.
137 .RE

139 .sp
140 .ne 2
141 .na
142 \fB\fBPOE_SYSTEM\fR\fR
143 .ad
144 .RS 20n
145 There is not enough memory available to allocate working buffers. Check
146 \fBerrno\fR for the specific system error code.
147 .RE

149 .SH EXAMPLES
157 .LP
150 \fBExample 1 \fRRetrieve the pool named "foo" from a given configuration.
151 .sp
152 .in +2
153 .nf
154 #include <pool.h>
155 #include <stdio.h>

157 \&...

159 pool_conf_t *conf;
160 pool_t *pool;

162 \&...

164 if ((pool = pool_get_pool(conf, "foo")) == NULL) {
165         (void) fprintf(stderr, "Cannot retrieve pool named
166         'foo'\\n");
174         'foo'\\B{}n");
167         ...
168 }
169 .fi
170 .in -2

172 .SH ATTRIBUTES
181 .sp
182 .LP
173 See \fBattributes\fR(5) for descriptions of the following attributes:
174 .sp

176 .sp
177 .TS
178 box;
```



```
179 c | c
180 l | l .
181 ATTRIBUTE TYPE  ATTRIBUTE VALUE
182 _
183 CSI     Enabled
184 _
185 Interface Stability     Unstable
186 _
187 MT-Level        Safe
188 .TE

190 .SH SEE ALSO
201 .sp
202 .LP
191 \fBlibpool\fR(3LIB), \fBpool_error\fR(3POOL), \fBattributes\fR(5)
```

```
**********************************************************
    8407 Sun Jan 19 01:29:09 2020
new/usr/src/man/man3pool/pool_resource_create.3pool
12202 noise in example code in some section 3pool man pages
**********************************************************
```

```
   1 '\" te
   2 .\" Copyright (c) 2003, Sun Microsystems, Inc. All Rights Reserved.
   3 .\" The contents of this file are subject to the terms of the Common Development
   4 .\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
   5 .\" When distributing Covered Code, include this CDDL HEADER in each file and in
   6 .TH POOL_RESOURCE_CREATE 3POOL "January 18, 2020"
   6 .TH POOL_RESOURCE_CREATE 3POOL "January 15, 2020"
   7 .SH NAME
   8 pool_resource_create, pool_resource_destroy, pool_resource_info,
   9 pool_query_resource_components, pool_resource_transfer, pool_resource_xtransfer
  10 \- resource pool resource manipulation functions
  11 .SH SYNOPSIS
  12 .nf
  13 cc [ \fIflag\fR\&.\|.\|. ] \fIfile\fR\&.\|.\|. \fB-lpool\fR [ \fIlibrary\fR\&.\|
  14 #include <pool.h>

  16 \fBpool_resource_t *\fR\fBpool_resource_create\fR(\fBpool_conf_t *\fR\fIconf\fR,
  17      \fBconst char *\fR\fItype\fR, \fBconst char *\fR\fIname\fR);
  18 .fi

  20 .LP
  21 .nf
  22 \fBint\fR \fBpool_resource_destroy\fR(\fBpool_conf_t *\fR\fIconf\fR,
  23      \fBpool_resource_t *\fR\fIresource\fR);
  24 .fi

  26 .LP
  27 .nf
  28 \fBconst char *\fR\fBpool_resource_info\fR(\fBpool_conf_t *\fR\fIconf\fR
  29      \fBpool_resource_t *\fR\fIresource\fR, \fBint\fR \fIflags\fR);
  30 .fi

  32 .LP
  33 .nf
  34 \fBpool_component_t **\fR\fBpool_query_resource_components\fR(
  35      \fBpool_conf_t *\fR\fIconf\fR, \fBpool_resource_t *\fR\fIresource\fR,
  36      \fBuint_t *\fR\fInelem\fR, \fBpool_value_t **\fR\fIprops\fR);
  37 .fi

  39 .LP
  40 .nf
  41 \fBint\fR \fBpool_resource_transfer\fR(\fBpool_conf_t *\fR\fIconf\fR,
  42      \fBpool_resource_t *\fR\fIsource\fR, \fBpool_resource_t *\fR\fItarget\fR,
  43      \fBuint64_t\fR \fIsize\fR);
  44 .fi

  46 .LP
  47 .nf
  48 \fBint\fR \fBpool_resource_xtransfer\fR(\fBpool_conf_t *\fR\fIconf\fR,
  49      \fBpool_resource_t *\fR\fIsource\fR, \fBpool_resource_t *\fR\fItarget\fR,
  50      \fBpool_component_t **\fR\fIcomponents\fR);
  51 .fi

  53 .SH DESCRIPTION
  54 The \fBpool_resource_create()\fR function creates and returns a new resource of
  55 the given \fIname\fR and \fItype\fR in the provided configuration. If there is
  56 already a resource of the given name, the operation will fail.
  57 .sp
  58 .LP
  59 The \fBpool_resource_destroy()\fR function removes the specified \fIresource\fR
  60 from its configuration file.
```



```
  61 .sp
  62 .LP
  63 The \fBpool_resource_info()\fR function returns a string describing the given
  64 \fIresource\fR. The string is allocated with \fBmalloc\fR(3C). The caller is
  65 responsible for freeing the returned string. If the \fIflags\fR argument is
  66 non-zero, the string returned also describes the components (if any) contained
  67 in the resource.
  68 .sp
  69 .LP
  70 The \fBpool_query_resource_components()\fR function returns a null-terminated
  71 array of the components (if any) that comprise the given resource.
  72 .sp
  73 .LP
  74 The \fBpool_resource_transfer()\fR function transfers \fIsize\fR basic units
  75 from the \fIsource\fR resource to the \fItarget\fR. Both resources must be of
  76 the same type for the operation to succeed. Transferring component resources,
  77 such as processors, is always performed as series of
  78 \fBpool_resource_xtransfer()\fR operations, since discrete resources must be
  79 identified for transfer.
  80 .sp
  81 .LP
  82 The \fBpool_resource_xtransfer()\fR function transfers the specific
  83 \fIcomponents\fR from the \fIsource\fR resource to the \fItarget\fR. Both
  84 resources must be of the same type, and of a type that contains components
  85 (such as processor sets). The \fIcomponents\fR argument is a null-terminated
  86 list of \fBpool_component_t\fR.
  87 .sp
  88 .LP
  89 The \fIconf\fR argument for each function refers to the target configuration to
  90 which the operation applies.
  91 .SH RETURN VALUES
  92 Upon successful completion, \fBpool_resource_create()\fR returns a new
  93 \fBpool_resource_t\fR with default properties initialized. Otherwise,
  94 \fINULL\fR is returned and \fBpool_error\fR(3POOL) returns the pool-specific
  95 error value.
  96 .sp
  97 .LP
  98 Upon successful completion, \fBpool_resource_destroy()\fR returns 0. Otherwise,
  99 -1 is returned and \fBpool_error()\fR returns the pool-specific error value.
 100 .sp
 101 .LP
 102 Upon successful completion, \fBpool_resource_info()\fR returns a string
 103 describing the given resource (and optionally its components). Otherwise,
 104 \fINULL\fR is returned and \fBpool_error()\fR returns the pool-specific error
 105 value.
 106 .sp
 107 .LP
 108 Upon successful completion, \fBpool_query_resource_components()\fR returns a
 109 null-terminated array of \fBpool_component_t *\fR that match the provided
 110 null-terminated property list and are contained in the given resource.
 111 Otherwise, \fINULL\fR is returned and \fBpool_error()\fR returns the
 112 pool-specific error value.
 113 .sp
 114 .LP
 115 Upon successful completion, \fBpool_resource_transfer()\fR and
 116 \fBpool_resource_xtransfer()\fR return 0. Otherwise -1 is returned and
 117 \fBpool_error()\fR returns the pool-specific error value.
 118 .SH ERRORS
 119 The \fBpool_resource_create()\fR function will fail if:
 120 .sp
 121 .ne 2
 122 .na
 123 \fB\fBPOE_BADPARAM\fR\fR
 124 .ad
 125 .RS 20n
 126 The supplied configuration's status is not \fBPOF_VALID\fR or \fIname\fR is in
```

```
127 use for this resource type.
128 .RE

130 .sp
131 .ne 2
132 .na
133 \fB\fBPOE_INVALID_CONF\fR\fR
134 .ad
135 .RS 20n
136 The resource element could not be created because the configuration would be
137 invalid.
138 .RE

140 .sp
141 .ne 2
142 .na
143 \fB\fBPOE_PUTPROP\fR\fR
144 .ad
145 .RS 20n
146 One of the supplied properties could not be set.
147 .RE

149 .sp
150 .ne 2
151 .na
152 \fB\fBPOE_SYSTEM\fR\fR
153 .ad
154 .RS 20n
155  A system error has occurred. Check the system error code for more details.
156 .RE

158 .sp
159 .LP
160 The \fBpool_resource_destroy()\fR function will fail if:
161 .sp
162 .ne 2
163 .na
164 \fB\fBPOE_BADPARAM\fR\fR
165 .ad
166 .RS 16n
167 The supplied configuration’s status is not \fBPOF_VALID\fR.
168 .RE

170 .sp
171 .LP
172 The \fBpool_resource_info()\fR function will fail if:
173 .sp
174 .ne 2
175 .na
176 \fB\fBPOE_BADPARAM\fR\fR
177 .ad
178 .RS 20n
179 The supplied configuration’s status is not \fBPOF_VALID\fR or the \fIflags\fR
180 parameter is neither 0 nor 1.
181 .RE

183 .sp
184 .ne 2
185 .na
186 \fB\fBPOE_INVALID_CONF\fR\fR
187 .ad
188 .RS 20n
189 The configuration is invalid.
190 .RE

192 .sp
```



```
193 .ne 2
194 .na
195 \fB\fBPOE_SYSTEM\fR\fR
196 .ad
197 .RS 20n
198 A system error has occurred. Check the system error code for more details.
199 .RE

201 .sp
202 .LP
203 The \fBpool_query_resource_components()\fR function will fail if:
204 .sp
205 .ne 2
206 .na
207 \fB\fBPOE_BADPARAM\fR\fR
208 .ad
209 .RS 20n
210 The supplied configuration’s status is not \fBPOF_VALID\fR.
211 .RE

213 .sp
214 .ne 2
215 .na
216 \fB\fBPOE_INVALID_CONF\fR\fR
217 .ad
218 .RS 20n
219 The configuration is invalid.
220 .RE

222 .sp
223 .ne 2
224 .na
225 \fB\fBPOE_SYSTEM\fR\fR
226 .ad
227 .RS 20n
228 A system error has occurred. Check the system error code for more details.
229 .RE

231 .sp
232 .LP
233 The \fBpool_resource_transfer()\fR function will fail if:
234 .sp
235 .ne 2
236 .na
237 \fB\fBPOE_BADPARAM\fR\fR
238 .ad
239 .RS 16n
240 The supplied configuration’s status is not \fBPOF_VALID\fR, the two resources
241 are not of the same type, or the transfer would cause either of the resources
242 to exceed their \fBmin\fR and \fBmax\fR properties.
243 .RE

245 .sp
246 .ne 2
247 .na
248 \fB\fBPOE_SYSTEM\fR\fR
249 .ad
250 .RS 16n
251 A system error has occurred. Check the system error code for more details.
252 .RE

254 .sp
255 .LP
256 The \fBpool_resource_xtransfer()\fR function will fail if:
257 .sp
258 .ne 2
```

```
 259 .na
 260 \fB\fBPOE_BADPARAM\fR\fR
 261 .ad
 262 .RS 20n
 263 The supplied configuration’s status is not \fBPOF_VALID\fR, the two resources
 264 are not of the same type, or the supplied resources do not belong to the
 265 source.
 266 .RE

 268 .sp
 269 .ne 2
 270 .na
 271 \fB\fBPOE_INVALID_CONF\fR\fR
 272 .ad
 273 .RS 20n
 274 The transfer operation failed and the configuration may be invalid.
 275 .RE

 277 .sp
 278 .ne 2
 279 .na
 280 \fB\fBPOE_SYSTEM\fR\fR
 281 .ad
 282 .RS 20n
 283 A system error has occurred. Check the system error code for more details.
 284 .RE

 286 .SH EXAMPLES
 287 \fBExample 1 \fRCreate a new resource of type \fBpset\fR named \fBfoo\fR.
 288 .sp
 289 .in +2
 290 .nf
 291 #include <pool.h>
 292 #include <stdio.h>

 294 \&...

 296 pool_conf_t *conf;
 297 pool_resource_t *resource;
 298 \&...

 300 if ((resource = pool_resource_create(conf, "pset",
 301     "foo")) == NULL) {
 302     (void) fprintf(stderr, "Cannot create resource\\n");
 302     (void) fprintf(stderr, "Cannot create resource\\B{}n");
 303     ...
 304 }
```

_____*unchanged_portion_omitted_*