DEOS-FY2013-SP-01E

D-RE APIs Sample Programs Manual

Version E1.02

2013/09/01

Edited by DEOS R&D Center

DEOS Project

JST-CREST

Research Area

"Dependable Operating Systems for Embedded Systems Aiming at Practical Applications"

Japan Science and Technology Agency

Contents

History

Version	変更内容	変更者	日付
1.02	Corrected the omittions and typo	Dependable Embedded OS R&D Center	2013.Sep.1

1 Sample programs which call the APIs of the Containers

1.1 Sample programs which call the APIs of the Application Containers

1.1.1How to use

When libdre0-dev is installed, the source codes for the sample programs that use these APIs will be installed in the following path.

/usr/share/doc/libdre0-dev/sample/app

Run "make" in the above path to compile the samples.

1.1.2Description of the sample programs

The following samples are available.

dre_app_bind_pid.c
Program for binding a process in the Application Container into the Control Group (cgroup)

dre_app_commit_snapshot.c
Program for creating a new Base Image of the Application Container from a snapshot

dre_app_create_cgroup.c
Program for creating a Control Group in the Application Container

dre_app_create_container.c
Program for creating an Application Container

dre_app_destroy_cgroup.c
Program for removing a Control Group on the Application Container

dre_app_destroy_container.c
Program for removing an Application Container

dre_app_get_base_list.c
Program for listing the Base Images (filesystem) of the Application Containers

dre_app_get_container_list.c
Program for listing the Application Containers in the system

dre_app_get_registered_pid_list.c
Program for selecting/listing PIDs from "/proc/daware" for the Application Container

dre_app_get_resource.c
Program for obtaining the information of resource usage of an Application Container

dre_app_get_snapshot_list.c
Program for listing the snapshots of an Application Container

dre_app_get_status
Program for getting the status of an Application Container

dre_app_load_snapshot.c
Program for restoring the file system of an Application Container

dre_app_remove_base.c
Program for removing a Base Image of the Application Containers

dre_app_remove_snapshot.c
Program for removing a snapshot of an Application Container

dre_app_resume.c
Program for resuming an Application Container which is suspended

dre_app_save_snapshot.c
Program for generating a snapshot of an Application Container
dre_app_set_resource.c
Program for setting limitation on resource usage of an Application Container
dre_app_start.c
Program for starting an Application Container
dre_app_stop.c
Program for stopping an Application Container
dre_app_suspend.c
Program for suspending an Application Container which is running
dre_app_unbind_pid.c
Program for unbinding a process which is bound to the Control Group in an Application Container
dre_app_unset_resource.c
Program for releasing the limitation of resource usage of an Application Container

1.1.2.1 dre_app_bind_pid.c

Summary

Program for binding a process in the Application Container into the Control Group (cgroup)
(Corresponds to "dre-app bind [-n <name>] -g <cgroup> -pid <pid>")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
10          char *cgroup = NULL;
            . . . (snip) . . .
27          /* open handle */
29          h = dre_app_open(name);
30          if(!h) {
31                  result = DRE_ERR_NULL_POINTER;
32                  goto error;
34          }
35
36          /* register pid */
37          result = dre_app_register_pid(h, pid);
38          if(result) goto error;
39
40          /* bind to control group */
41          result = dre_app_bind_pid(h, cgroup, pid);
42          if(result) goto error;
43
44          /* unregister pid */
```

```
45          result = dre_app_unregister_pid(h, pid);
46          if(result) goto error;
47
48          /* close handle */
50          result = dre_app_close(h);
51          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_bind_pid.c" for the full text of the source code.)

In Line 29, "dre_app_open()" gets the handle of "<name>", if the name of the Application Container is specified as "<name>". In Line 37, "dre_app_register_pid()" correlates the process ID assigned by the Application Container called VPID to the process ID assigned by the Host OS called PID. In Line 41, "dre_app_bind_pid()" binds "<pid>" to the Control Group specified as "<cgroup>". At this time, "<cgroup>" means the Control Group in the "<name>" if "<name>" is specified. If "<name>" is not specified, "<cgroup>" means the Control Group on the Host OS. In Line 45, "dre_app_unregister_pid()" breaks the correlation of PID and VPID. In Line 50, "dre_app_close()" releases the handle of "<name>", if "<name>" is specified.

Usage

dre_app_bind_pid -g <cgroup> --pid <pid> [-n <name>]

Arguments

<cgroup>	the name of the Control Group
<pid>	process ID
<name>	the name of the Application Container (optional)

1.1.2.2 dre_app_commit_snapshot.c

Summary

Program for creating a new Base Image of the Application Container from a snapshot
(Corresponds to "dre-app commit -n <name> -t <tag> -b <base_image>")

Description

```
3   #include "dre_app.h"
4
5   int main(int argc, char **argv)
6   {
7           dre_return_t result:
8           dre_app_handle *h = NULL;
9           char *name = NULL;
10          char *tag = NULL;
11          char *base = NULL;
            . . . (snip) . . .
27          /* open handle */
28          h = dre_app_open(name);
29          if(!h) {
30                  result = DRE_ERR_NULL_POINTER;
31                  goto error;
```

```
32          }
33
34          /* commit snapshot */
35          result = dre_app_commit_snapshot(h, snapshot, base);
36          if(result) goto error;
37
38          /* close handle */
39          result = dre_app_close(h);
40          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_commit_snapshot.c" for the full text of the source code.)

In Line 28, "dre_app_open()" gets the handle of "<name>". In Line 35, "dre_app_commit_snapshot()" creates a new Base Image, "<base_image>", from the snapshot named "<tag>" of the Application Container named "<name>". In Line 39, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_commit_snapshot -n <name> -t <tag> -b <base_image>

Arguments

<name>	the name of the Application Container
<tag>	the name of the snapshot
<base_image>	the name of the Base Image to create

1.1.2.3 dre_app_create_cgroup.c

Summary

Program for creating a Control Group in the Application Container
(Corresponds to "dre-app create [-n <name>] -g <cgroup>")

Description

```
3   #include "dre_app.h"
4
5   int main(int argc, char **argv)
6   {
7           dre_return_t result;
8           dre_app_handle *h = NULL;
9           char *name = NULL;
10          char *cgroup = NULL;
            . . . (snip) . . .
24          /* open handle */
25          if(name) {
26                  h = dre_app_open(name);
27                  if(!h) {
28                          result = DRE_ERR_NULL_POINTER;
29                          goto error;
30                  }
```

```
31      }
32
33      /* create control group */
34      result = dre_app_create_cgroup(h, cgroup);
35      if(result) goto error;
36
37      /* close handle */
38      if(h) {
39              result = dre_app_close(h);
40              if(result) goto error;
41      }
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_create_cgroup.c" for the full text of the source code.)

In Line 26, "dre_app_open()" gets the handle of "<name>", if the name of the Application Container is specified as "<name>". In Line 34, "dre_app_create_cgroup()" generates a Control Group named "<cgroup>". If "<name>" is specified, "<cgroup>" is generated in "<name>". Otherwise, "<cgroup>" is generated in the Host OS. In Line 39, "dre_app_close()" releases the handle of "<name>", if "<name>" is specified.

Usage

dre_app_create_cgroup -g <cgroup> [-n <name>]

Arguments

<cgroup>	the name of the Control Group to create
<name>	the name of the Application Container (optional)

1.1.2.4 dre_app_create_container.c

Summary

Program for creating an Application Container
(Corresponds to "dre-app create -n <name>")

Description

```
3   #include "dre_app.h"
4
5   int main(int argc, char **argv)
6   {
7       dre_return_t result;
8       char *name = NULL;
        . . . (snip) . . .
20      /* create container */
21      dre_app_data data = {DRE_APP_CONTAINER_TYPE_LXC, 2200, NULL};
22      result = dre_app_create_container(name, &data);
23      if(result) goto error;{
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_create_container.c" for the full text of the source code.)

In Line 21, the setting values of "<name>" are specified. In the above example, the setting values of "<name>" are the following.

- Use a Container of LXC
- Use the port 2200 for SSH
- Use the default Base Image

In Line 22, "dre_app_create_container" creates the Application Container using the value of "<name> (specified by the arguments)" and the setting specified In Line 15.

Usage

dre_app_create_container -n <name>

Arguments

<name> the name of the Application Container

1.1.2.5 dre_app_destroy_cgroup.c

Summary

Program for removing a Control Group on the Application Container
(Corresponds to "dre-app destroy [-n <name>] -g <cgroup>")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
10          char *cgroup = NULL;
            . . . (snip) . . .
24          /* open handle */
25          if(name) {
26                  h = dre_app_open(name);
27                  if(!h) {
28                          result = DRE_ERR_NULL_POINTER;
29                          goto error;
30                  }
31          }
32
33          /* destroy control group */
34          result = dre_app_destroy_cgroup(h, cgroup);
35          if(result) goto error;
36
37          /* close handle */
38          if(h) {
39                  result = dre_app_close(h);
40                  if(result) goto error;
```

```
41      }
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_destroy_cgroup.c" for the full text of the source code.)

In Line 26, "dre_app_open()" gets the handle of "<name>", if the name of the Application Container is specified as "<name>". In Line 34, "dre_app_destroy_cgroup()" removes a Control Group named "<cgroup>". If "<name>" is specified, "<cgroup>" in "<name> is removed". Otherwise, "<cgroup>" in the Host OS is removed. In Line 39, "dre_app_close()" releases the handle of "<name>", if "<name>" is specified.

Usage

dre_app_destroy_cgroup -g <cgroup> [-n <name>]

Arguments

<cgroup>	the name of the Control Group
<name>	the name of the Application Container (optional)

1.1.2.6 dre_app_destroy_container.c

Summary

Program for removing an Application Container
(Corresponds to "dre-app destroy -n <name>")

Description

```
3   #include "dre_app.h"
4
5   int main(int argc, char **argv)
6   {
7       dre_return_t result;
8       char *name = NULL;
        . . . (snip) . . .
20      /* destroy container */
21      result = dre_app_destroy_container(name);
22      if(result) goto error;{
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_destroy_container.c" for the full text of the source code.)

In Line 21, "dre_app_detroy_container()" removes the Application Container named "<name>".

Usage

dre_app_destroy_container -n <name>

Arguments

<name>	the name of the Application Container

1.1.2.7 dre_app_get_base_list.c

Summary

Program for listing the Base Images (filesystem) of the Application Containers
(Corresponds to "dre-app list --base")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_list_t list;
            . . . (snip) . . .
14          /* get base image list */
15          result = dre_app_get_base_list(&list);
16          if(result) goto error;
17          if(list) {
18                  while(*(list + i)) {
19                          printf("%s¥n", *(list + i));
20                          i++;
21                  }
22          }
23
24          /* release base image list */
25          if(list) {
26                  result = dre_app_release_base_list(list);
27                  if(result) goto error;
28          }
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_get_base_list.c" for the full text of the source code.)

In Line 15, "dre_app_get_base_list()" gets the list of the Base Images in the system. In Lines 18 to 21, the names of the Base Images in the list are enumerated. In Line 26, "dre_app_release_base_list()" releases the list of the name of the Base Images.

Usage

dre_app_get_base_list

Arguments

None

1.1.2.8 dre_app_get_container_list.c

Summary

Program for listing the Application Containers in the system
(Corresponds to "dre-app list")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_list_t list;
            . . . (snip) . . .
14          /* get container list */
15          result = dre_app_get_container_list(&list);
16          if(result) goto error;
17          if(list) {
18                  while(*(list + i)) {
19                          printf("%s¥n", *(list + i));
20                          i++;
21                  }
22          }
23
24          /* release container list */
25          if(list) {
26                  result = dre_app_release_container_list(list);
27                  if(result) goto error;
28          }
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/list.c" for the full text of the source code.)

In Line 15, "dre_app_get_container_list()" gets the list of the name of the Application Containers on the system. In Lines 18 to 21, the names of the Application Containers in the list are enumerated. In Line 26, "dre_app_release_container_list()" releases the list of the name of the Application Containers.

Usage

dre_app_get_container_list

Arguments

None

1.1.2.9 dre_app_get_registered_pid_list.c

Summary

Program for selecting/listing PIDs from "/proc/daware" for the Application Container
(Corresponds to "dre-app pidlist -n <name>")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
```

```
 8        dre_app_handle *h = NULL;
 9        dre_app_pid_list_t plist;
10        char *name = NULL;
          . . . (snip) . . .
22        /* open handle */
23        h = dre_app_open(name);
24        if(!h) {
25              result = DRE_ERR_NULL_POINTER;
26              goto error;
27        }
28
29        /* get pid list */
30        result = dre_app_get_registered_pid_list(h, &plist);
31        if(result) goto error;
32        if(plist) {
33              int j = 0;
34              printf("Registered process(es) on  '%s':¥n", name);
35              while(plist[j]) {
36                    printf("- pid:%d(vpid:%d)¥n", (*plist[j]).pid, (*plist[j]).vpid);
37                    j++;
38              }
39        } else {
40              printf("No process registered on  '%s':¥n", name);
41        }
42
43        /* release pid list */
44        if(list) {
45              result = dre_app_release_registered_pid_list(list);
46              if(result) goto error;
47        }
48
49        /* close handle */
50        result = dre_app_close(h);
51        if(result) goto error;
          . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_get_container_list.c" for the full text of the source code.)

In Line 23, "dre_app_open()" gets the handle of "<name>". In Line 30, "dre_app_get_registered_pid_list()" gets the list of the PIDs registered to "/proc/daware". In Lines 33 to 38, the PIDs in the list are enumerated. In Line 45, "dre_app_release_registered_pid_list()" releases the list of the PIDs. In Line 50, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_get_registered_pid_list -n <name>

Arguments

<name>	the name of the Application Container

1.1.2.10 dre_app_get_resource.c

Summary

Program for obtaining the information of resource usage of an Application Container
(Corresponds to "dre-app usage -n <name> -pid <pid> [-g <cgroup>]")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_app_resource_value_t resource;
 8          dre_return_t result:
 9          dre_app_handle *h = NULL;
10          char *name = NULL;
11          char *type = NULL;
            . . . (snip) . . .
37          /* open handle */
38          h = dre_app_open(name);
39          if(!h) {
40                  result = DRE_ERR_NULL_POINTER;
41                  goto error;
42          }
43
44          /* register process */
45          if(pid > 0) {
46                  result = dre_app_register_pid(h, pid);
47                  if(result) goto error;
48          }
49
50          /* get resource usage */
51          if(!strcmp(type, "cpu")) {
52                  resource.type = DRE_APP_CONTAINER_RESOURCE_CPU;
53                  result = dre_app_get_resource(h, pid, &resource);
54                  if(result) goto error;
55                  printf("cpu usage:       %d%%¥n", resource.udata.cpu.percent);
56          } else if(!strcmp(type, "core")) {
57                  resource.type = DRE_APP_CONTAINER_RESOURCE_CPU_CORE;
58                  result = dre_app_get_resource(h, pid, &resource);
59                  if(result) goto error;
60                  printf("assined cpu core(s):       %s¥n", resource.udata.cpu.cores);
61          } else if(!strcmp(type, "mem")) {
62                  resource.type = DRE_APP_CONTAINER_RESOURCE_MEM;
63                  result = dre_app_get_resource(h, pid, &resource);
64                  if(result) goto error;
65                  printf("memory usage: %dBytes¥n", resource.udata.mem.usage_in_bytes);
66          } else if(!strcmp(type, "memsw")) {
67                  resource.type = DRE_APP_CONTAINER_RESOURCE_MEMSW;
```

```
68              result = dre_app_get_resource(h, pid, &resource);
69              if(result) goto error;
70              printf("memory+swap usage: %dBytes¥n", resource.udata.mem.memsw_usage_in_bytes);
71      } else {
72              printf("Invalid resource type: %s¥n", type);
73              result = DRE_ERR_INVALID_ARGUMENT;
74              goto error;
75      }
76
77      /* unregister process */
78      if(pid > 0) {
79              result = dre_app_unregister_pid(h, pid);
80              if(result) goto error;
81      }
82
83      /* close handle */
84      result = dre_app_close(h);
85      if(result) goto error;
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_get_resource.c" for the full text of the source code.)

In Line 38, "dre_app_open()" gets the handle of "<name>". In Line 46, "dre_app_register_pid()" correlates the process ID assigned by the Application Container called VPID to the process ID assigned by the Host OS called PID. In Lines 51 to 75, "dre_app_get_resource()" obtains the information of the resource usage and outputs it.
"dre_app_get_resource()" works as follows.

- It obtains the information of the resource usage of the "<cgroup>" in the "<name>", if "<cgroup>" is specified.
  - In addition to the above, it obtains the information of the resource usage of "<pid>" which is bound to "<cgroup>", if "<pid>" is specified.
- It obtains the information of the resource usage of the "<name>", if "<cgroup>" is omitted.
  - In addition to above, it obtains the information of the resource usage of "<pid>" in "<name>", if "<pid>" is specified.

In Line 79, "dre_app_unregister_pid()" breaks the correlation of PID and VPID, if "<pid>" is specified. In Line 80, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_get_resource -n <name> <resource_type> [--pid <pid>] [-g <cgroup>]

Arguments

<name>	the name of the Application Container
<resource_type>	type of the resource
	"cpu" : CPU usage (%)
	"core" : CPU core(s)
	"mem" : Memory usage (bytes)
	"memsw" : Memory and Swap usage (bytes)
<pid>	process ID
<cgroup>	the name of the Control Group (optional)

1.1.2.11 dre_app_get_snapshot_list.c

Summary

Program for listing the snapshots of an Application Container
(Corresponds to "dre-app list --snapshot -n <name>")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_app_handle *h = NULL;
 9          dre_list_t list;
10          char *name = NULL;
            . . . (snip) . . .
22          /* open handle */
23          h = dre_app_open(name);
24          if(!h) {
25                  result = DRE_ERR_NULL_POINTER;
26                  goto error;
27          }
28
29          /* get snapshot list */
30          result = dre_app_get_snapshot_list(h, &list);
31          if(result) goto error;
32          if(list) {
33                  int j = 0;
34                  while(*(list + j)) {
35                          printf("%s¥n", *(list + j));
36                          j++;
37                  }
38          }
39
40          /* release snapshot list */
41          if(list) {
42                  result = dre_app_release_snapshot_list(list);
43                  if(result) goto error;
44          }
45
46          /* close handle */
47          result = dre_app_close(h);
48          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_get_snapshot_list.c" for the full text of the source code.)

In Line 23, "dre_app_open()" gets the handle of "<name>". In Line 30, "dre_app_get_snapshot_list()" gets a list of the names of the snapshots. In Lines 33 to 37, the names of the snapshots in the list are enumerated. In Line 42,

"dre_app_release_snapshot_list()" releases the list of the snapshots. In Line 47, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_get_snapshot_list -n <name>

Arguments

<name> the name of the Application Container

1.1.2.12 dre_app_get_status

Summary

Program for getting the status of an Application Container
(Corresponds to "dre-app status -n <name>")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
            . . . (snip) . . .
21          /* open handle */
22          h = dre_app_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* get status */
29          result = dre_app_get_status(h);
30          if(result == DRE_APP_CONTAINER_STATUS_STOP) {
31                  printf("'%s' is STOPPED.¥n", name);
32          } else if(result == DRE_APP_CONTAINER_STATUS_RUNNING) {
33                  printf("'%s' is RUNNING.¥n", name);
34          } else if(result == DRE_APP_CONTAINER_STATUS_SUSPENDED) {
35                  printf("'%s' is SUSPENDED.¥n", name);
36          } else if(result == DRE_APP_CONTAINER_STATUS_BUSY) {
37                  printf("'%s' is BUSY.¥n", name);
38          } else {
39                  result = DRE_ERR_CONTAINER_UNKNOWN_STATUS;
40                  goto error;
41          }
42
43          /* close handle */
```

```
44      result = dre_app_close(h);
45      if(result) goto error;
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_get_status.c" for the full text of the source code.)

In Line 22, "dre_app_open()" gets the handle of "<name>". In Line 29, "dre_app_get_status()" gets the status of "<name>". In Lines 30 to 41, the result of "dre_app_get_status()" is output. In Line 44, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_get_status -n <name>

Arguments

<name>	the name of the Application Container

1.1.2.13 dre_app_load_snapshot.c

Summary

Program for restoring the file system of an Application Container
(Corresponds to "dre-app load -n <name> -t <tag>")

Description

```
3   #include "dre_app.h"
4
5   int main(int argc, char **argv)
6   {
7       dre_return_t result:
8       dre_app_handle *h = NULL;
9       char *name = NULL;
10      char *snapshot = NULL;
        . . . (snip) . . .
24      /* open handle */
25      h = dre_app_open(name);
26      if(!h) {
27          result = DRE_ERR_NULL_POINTER;
28          goto error;
29      }
30
31      /* load snapshot */
32      result = dre_app_load_snapshot(h, snapshot);
33      if(result) goto error;
34
35      /* close handle */
36      result = dre_app_close(h);
37      if(result) goto error;
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_load_snapshot.c" for the full text of the source code.)

In Line 25, "dre_app_open()" gets the handle of "<name>". In Line 32, "dre_app_load_snapshot()" restores the file system of the Application Container named "<name>" with the snapshot named "<tag>". In Line 36, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_load_snapshot -n <name> -t <tag>

Arguments

<name>	the name of the Application Container
<tag>	the name of the snapshot

1.1.2.14 dre_app_remove_base.c

Summary

Program for removing a Base Image of the Application Containers
(Corresponds to "dre-app remove --base <base_image>")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_app_handle *h = NULL;
 9          char *base = NULL;
            . . . (snip) . . .
21          /* remove base image */
22          result = dre_app_remove_base(base);
23          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_remove_base.c" for the full text of the source code.)

In Line 22, "dre_app_remove_base()" removes the Base Image named "<name>".

Usage

dre_app_remove_base -b <base_image>

Arguments

<base_image>	the name of the Base Image

1.1.2.15 dre_app_remove_snapshot.c

Summary

Program for removing a snapshot of an Application Container

(Corresponds to "dre-app remove --snapshot -n <name> -t <tag>")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
10          char *snapshot = NULL;
            . . . (snip) . . .
24          /* open handle */
25          h = dre_app_open(name);
26          if(!h) {
27                  result = DRE_ERR_NULL_POINTER;
28                  goto error;
29          }
30
31          /* remove snapshot */
32          result = dre_app_remove_snapshot(h, snapshot);
33          if(result) goto error;
34
35          /* close handle */
36          result = dre_app_close(h);
37          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_remove_snapshot.c" for the full text of the source code.)

In Line 25, "dre_app_open()" gets the handle of "<name>". In Line 32, "dre_app_remove_snapshot()" removes the snapshot named "<tag>" of the Application Container named "<name>". In Line 36, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_remove_snapshot -n <name> -t <tag>

Arguments

<name>	the name of the Application Container
<tag>	the name of the snapshot

1.1.2.16 dre_app_resume.c

Summary

Program for resuming an Application Container which is suspended
(Corresponds to "dre-app resume -n <name>")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
            . . . (snip) . . .
21          /* open handle */
22          h = dre_app_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* resume container */
29          result = dre_app_resume(h);
30          if(result) goto error;
31
32          /* close handle */
33          result = dre_app_close(h);
34          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_resume.c" for the full text of the source code.)

In Line 22, "dre_app_open()" gets the handle of "<name>". In Line 29, "dre_app_resume()" resumes the Application Container named "<name>" which was suspended. In Line 33, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_resume -n <name>

Arguments

<name> the name of the Application Container

1.1.2.17 dre_app_save_snapshot.c

Summary

Program for generating a snapshot of an Application Container
(Corresponds to "dre-app save -n <name> -t <tag>")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
```

```
 9        char *name = NULL;
10        char *snapshot = NULL;
          . . . (snip) . . .
24        /* open handle */
25        h = dre_app_open(name);
26        if(!h) {
27              result = DRE_ERR_NULL_POINTER;
28              goto error;
29        }
30
31        /* save snapshot */
32        result = dre_app_save_snapshot(h, snapshot);
33        if(result) goto error;
34
35        /* close handle */
36        result = dre_app_close(h);
37        if(result) goto error;
          . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_save_snapshot.c" for the full text of the source code.)

In Line 25, "dre_app_open()" gets the handle of "<name>". In Line 32, "dre_app_save_snapshot()" generates a snapshot of the file system of the Application Container named "<name>" as "<tag>". In Line 36, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_save_snapshot -n <name> -t <tag>

Arguments

<name>	the name of the Application Container
<tag>	the name of the snapshot

1.1.2.18 dre_app_set_resource.c

Summary

Program for setting limitation on resource usage of an Application Container
(Corresponds to "dre-app limit <resource_type> <limitation> -n <name> [-g <cgroup>]")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7        dre_return_t result:
 8        dre_app_handle *h = NULL;
 9        dre_app_resource_value_t resource;
10        char *name = NULL;
11        char *cgroup = NULL;
```

```
12      char *type = NULL;
13      char *limitation = NULL;
        . . . (snip) . . .
39      /* set resource usage */
40      if(!strcmp(type, "cpu")) {
41              resource.type = DRE_APP_CONTAINER_RESOURCE_CPU;
42              resource.udata.cpu.percent = atoi(limitation);
43      } else if(!strcmp(type, "core")) {
44              resource.type = DRE_APP_CONTAINER_RESOURCE_CPU_CORE;
45              resource.udata.cpu.cores = limitation;
46      } else if(!strcmp(type, "mem")) {
47              resource.type = DRE_APP_CONTAINER_RESOURCE_MEM;
48              resource.udata.mem.usage_in_bytes = atoi(limitation);
49      } else if(!strcmp(type, "memsw")) {
50              resource.type = DRE_APP_CONTAINER_RESOURCE_MEMSW;
51              resource.udata.mem.memsw_usage_in_bytes = atoi(limitation);
52      } else {
53              printf("Invalid resource type: %s¥n", type);
54              result = DRE_ERR_INVALID_ARGUMENT;
55              goto error;
56      }
57
58      /* open handle */
59      if(name) {
60              h = dre_app_open(name);
61              if(!h) {
62                      result = DRE_ERR_NULL_POINTER;
63                      goto error;
64              }
65      }
66
67      /* set the limitation */
68      result = dre_app_set_resource(h, cgroup, resource);
69      if(result) goto error;
70      printf("Set the limitation of %s.¥n", type);
71
72      /* close handle */
73      if(h) {
74              result = dre_app_close(h);
75              if(result) goto error;
76      }
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_set_resource.c" for the full text of the source code.)

In Lines 40 to 56 specify the resources and the limitation of their usage. These lines work as follows.

- It sets the resource usage of the "<cgroup>" in the "<name>", if "<cgroup>" is specified.
- It sets the resource usage of the "<name>", if "<cgroup>" is omitted.

In Line 60, "dre_app_open()" gets the handle of "<name>". In Line 68, "dre_app_set_resource()" sets the limitation on resource usage based on the result of the lines from 40 to 56. In Line 74, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_set_resource -n <name> <resource_type> <limitation> [-g <cgroup>]

Arguments

<name>	the name of the Application Container
<resource_type>	type of the resource
	"cpu" : CPU usage (%)
	"core" : CPU core(s)
	"mem" : Memory usage (bytes)
	"memsw" : Memory and Swap usage (bytes)
<limitation>	the limitation on resource usage
<cgroup>	the name of the Control Group (optional)

1.1.2.19 dre_app_start.c

Summary

Program for starting an Application Container
(Corresponds to "dre-app start -n <name>")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
            . . . (snip) . . .
21          /* open handle */
22          h = dre_app_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* start container */
29          result = dre_app_start(h);
30          if(result) goto error;
31
32          /* close handle */
33          result = dre_app_close(h);
34          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_start.c" for the full text of the source code.)

In Line 22, "dre_app_open()" gets the handle of "<name>". In Line 29, "dre_app_start()" starts the Application Container named "<name>". In Line 33, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_start -n <name>

Arguments

<name> the name of the Application Container

1.1.2.20 dre_app_stop.c

Summary

Program for stopping an Application Container
(Corresponds to "dre-app stop -n <name>")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_app_handle *h;
            . . . (snip) . . .
21          /* open handle */
22          h = dre_app_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* stop container */
29          result = dre_app_stop(h);
30          if(result) goto error;
31
32          /* close handle */
33          result = dre_app_close(h);
34          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_stop.c" for the full text of the source code.)

In Line 22, "dre_app_open()" gets the handle of "<name>". In Line 29, "dre_app_stop()" stops the Application Container named "<name>" which is running. In Line 33, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_stop -n <name>

Arguments

<name> the name of the Application Container

1.1.2.21 dre_app_suspend.c

Summary

Program for suspending an Application Container which is running
(Corresponds to "dre-app suspend -n <name>")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
            . . . (snip) . . .
21          /* open handle */
22          h = dre_app_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* suspend container */
29          result = dre_app_suspend(h);
30          if(result) goto error;
31
32          /* close handle */
33          result = dre_app_close(h);
34          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_suspend.c" for the full text of the source code.)

In Line 22, "dre_app_open()" gets the handle of "<name>". In Line 29, "dre_app_suspend()" suspends the Application Container named "<name>" which is running. In Line 33, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_suspend -n <name>

Arguments

<name> the name of the Application Container

1.1.2.22 dre_app_unbind_pid.c

Summary

Program for unbinding a process which is bound to the Control Group in an Application Container (Corresponds to "dre-app unbind [-n <name>] -g <cgroup> -pid <pid>")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
10          char *cgroup = NULL;
            . . . (snip) . . .
27          /* open handle */
28          if(name) {
29                  h = dre_app_open(name);
30                  if(!h) {
31                          result = DRE_ERR_NULL_POINTER;
32                          goto error;
33                  }
34          }
35
36          /* register pid */
37          result = dre_app_register_pid(h, pid);
38          if(result) goto error;
39
40          /* unbind from control group */
41          result = dre_app_unbind_pid(h, cgroup, pid);
42          if(result) goto error;
43
44          /* unregister pid */
45          result = dre_app_unregister_pid(h, pid);
46          if(result) goto error;
47
48          /* close handle */
49          if(h) {
50                  result = dre_app_close(h);
51                  if(result) goto error;
52          }
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_unbind_pid.c" for the full text of the source code.)

In Line 29, "dre_app_open()" gets the handle of "<name>", if the name of the Application Container is specified as "<name>". In Line 37, "dre_app_register_pid()" correlates the process ID assigned by the Application Container called VPID to the process ID assigned by the Host OS, called PID. In Line 41, "dre_app_unbind_pid()" unbinds "<pid>" from the Control Group specified as "<cgroup>". At this time, "<cgroup>" means the Control Group in the "<name>" if "<name>" is specified. If "<name>" is not specified, "<cgroup>" means the Control Group on the Host OS. In Line 45,

"dre_app_unregister_pid()" breaks the correlation of PID and VPID. In Line 50, "dre_app_close()" releases the handle of "<name>", if "<name>" is specified.

Usage

dre_app_unbind_pid -g <cgroup> --pid <pid> [-n <name>]

Arguments

<cgroup>	the name of the Control Group
<pid>	process ID
<name>	the name of the Application Container (optional)

1.1.2.23 dre_app_unset_resource.c

Summary

Program for releasing the limitation of resource usage of an Application Container
(Corresponds to "dre-app unlimit <resource_type> -n <name> [-g <cgroup>]")

Description

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_app_resource_value_t resource;
 8          dre_return_t result:
 9          dre_app_handle *h = NULL;
10          char *name = NULL;
11          char *cgroup = NULL;
12          char *type = NULL;
            . . . (snip) . . .
34          /* unset resource usage */
35          if(!strcmp(type, "cpu")) {
36                  resource.type = DRE_APP_CONTAINER_RESOURCE_CPU;
37          } else if(!strcmp(type, "core")) {
38                  resource.type = DRE_APP_CONTAINER_RESOURCE_CPU_CORE;
39          } else if(!strcmp(type, "mem")) {
40                  resource.type = DRE_APP_CONTAINER_RESOURCE_MEM;
41          } else if(!strcmp(type, "memsw")) {
42                  resource.type = DRE_APP_CONTAINER_RESOURCE_MEMSW;
43          } else {
44                  printf("Invalid resource type: %s¥n", type);
45                  result = DRE_ERR_INVALID_ARGUMENT;
46                  goto error;
47          }
48
49          /* open handle */
50          if(name) {
51                  h = dre_app_open(name);
```

```
52              if(!h) {
53                      result = DRE_ERR_NULL_POINTER;
54                      goto error;
55              }
56      }
57
58      /* unset the limitation */
59      result = dre_app_unset_resource(h, cgroup, resource);
60      if(result) goto error;
61      printf("Unset the limitation of %s.¥n", type);
62
63      /* close handle */
64      if(h) {
65              result = dre_app_close(h);
66              if(result) goto error;
67      }
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/app/dre_app_unset_resource.c" for the full text of the source code.)

In Lines 35 to 47 specify the resource to unset the limitation of usage. These lines work as follows.

- They release the limitation on the resource usage of the "<cgroup>" in the "<name>", if "<cgroup>" is specified.
- They release the limitation of the resource usage of the "<name>", if "<cgroup>" is omitted.

In Line 51, "dre_app_open()" gets the handle of "<name>". In Line 59, "dre_app_unset_resource()" unsets the limitation on resource usage based on the result of the lines from 35 to 47. In Line 65, "dre_app_close()" releases the handle of "<name>".

Usage

dre_app_unset_resource -n <name> <resource_type> [-g <cgroup>]

Arguments

<name>	the name of the Application Container
<resource_type>	type of the resource
	"cpu" : CPU usage (%)
	"core" : CPU core(s)
	"mem" : Memory usage (bytes)
	"memsw" : Memory and Swap usage (bytes)
<cgroup>	the name of the Control Group (optional)

1.2 Sample programs which call the APIs of the System Containers

1.2.1How to use

When libdre0-dev is installed, the source codes for the sample programs that use these APIs will be installed in the following path.

/usr/share/doc/libdre0-dev/sample/sys

Run "make" in the above path to compile the samples.

1.2.2Description of the sample programs

The following samples are available.

dre_sys_bind_pid.c

Program for binding a process in the System Container into the Control Group

dre_sys_checkpoint.c

Program for generating a checkpoint of a System Container

dre_sys_commit_snapshot.c

Program for creating a new Base Image of the System Container from a snapshot

dre_sys_create_cgroup.c

Program for creating a Control Group in the System Container

dre_sys_create_container.c

Program for creating a System Container

dre_sys_destroy_cgroup.c

Program for removing a Control Group in the System Container

dre_sys_destroy_container.c

Program for removing a System Container

dre_sys_get_base_list.c

Program for listing the Base Images of the System Containers

dre_sys_get_checkpoint_list.c

Program for list the ckeckpoints of a System Container

dre_sys_get_container_list.c

Program for listing the System Containers on the system

dre_sys_get_resource.c

Program for obtaining the information of the resource usage of a System Container

dre_sys_get_snapshot_list.c

Program for listing the snapshots of a System Container

dre_sys_get_status.c

Program for getting the status of a System Container

dre_sys_immigrate.c

Program for starting a virtual machine of immigration in the target host in order to perform immigration of the System Container

dre_sys_load_snapshot.c

Program for restoring the file system of a System Container

dre_sys_migrate.c

Program for performing the migration of the System Container

dre_sys_remove_base.c

Program for removing a Base Image of the System Containers

dre_sys_remove_checkpoint.c
Program for removing a checkpoint of a System Container
dre_sys_remove_snapshot.c
Program for removing a snapshot of a System Container
dre_sys_restart.c
Program for restarting a System Container from a checkpoint
dre_sys_resume.c
Program for resuming a System Container which was suspended
dre_sys_save_snapshot.c
Program for generating a snapshot of a System Container
dre_sys_set_resource.c
Program for setting a limitation on resource usage of a System Container
dre_sys_share.c
Program for sharing the System Container with an NFS server
dre_sys_start.c
Program for starting a System Container
dre_sys_stop.c
Program for stopping a System Container
dre_sys_suspend.c
Program for suspending a System Container which is running
dre_sys_unbind_pid.c
Program for unbinding a process which is bound to the Control Group on a System Container
dre_sys_unset_resource.c
Program for releasing the limitation on resource usage of a System Container
dre_sys_unshare.c
Program for Program for stopping sharing the System Container between the Host OS and the NFS server

1.2.2.1 dre_sys_bind_pid.c

Summary

Program for binding a process in the System Container into the Control Group
(Corresponds to "dre-sys bind [-n <name>] -g <cgroup> -pid <pid>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
10          char *cgroup = NULL;
            . . . (snip) . . .
27          /* open handle */
28          if(name) {
29                  h = dre_sys_open(name);
```

```
30              if(!h) {
31                      result = DRE_ERR_NULL_POINTER;
32                      goto error;
33              }
34      }
35
36      /* bind to control group */
37      result = dre_sys_bind_pid(h, cgroup, pid);
38      if(result) goto error;
39
40      /* close handle */
41      if(h) {
42              result = dre_sys_close(h);
43              if(result) goto error;
44      }
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_bind_pid.c" for the full text of the source code.)

In Line 29, "dre_sys_open()" gets the handle of "<name>", if the name of the System Container is specified as "<name>". In Line 37, "dre_sys_bind_pid()" binds "<pid>" to the Control Group specified as "<cgroup>". At this time, "<cgroup>" means the Control Group in the "<name>" if "<name>" is specified. If "<name>" is not specified, "<cgroup>" means the Control Group in the Host OS. In Line 42, "dre_sys_close()" releases the handle of "<name>", if "<name>" is specified.

Usage

dre_sys_bind_pid -g <cgroup> --pid <pid> [-n <name>]

Arguments

<cgroup>	the name of the Control Group
<pid>	process ID
<name>	the name of the System Container (optional)

1.2.2.2 dre_sys_checkpoint.c

Summary

Program for generating a checkpoint of a System Container
(Corresponds to "dre-sys checkpoint -n <name> -t <tag>")

Description

```
3   #include "dre_sys.h"
4
5   int main(int argc, char **argv)
6   {
7           dre_return_t result:
8           dre_sys_handle *h = NULL;
9           char *name = NULL;
10          char *tag = NULL;
```

```
        . . . (snip) . . .
24      /* open handle */
25      h = dre_sys_open(name);
26      if(!h) {
27              result = DRE_ERR_NULL_POINTER;
28              goto error;
29      }
30
31      /* save checkpoint */
32      result = dre_sys_checkpoint(h, tag);
33      if(result) goto error;
34
35      /* close handle */
36      result = dre_sys_close(h);
37      if(result) goto error;
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_checkpoint.c" for the full text of the source code.)

In Line 25, "dre_sys_open()" gets the handle of "<name>". In Line 32, "dre_sys_checkpoint()" generates the checkpoint of the System Container named "<name>" as "<tag>". In Line 36, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_checkpoint -n <name> -t <tag>

Arguments

<name>	the name of the System Container
<tag>	the name of the checkpoint

1.2.2.3 dre_sys_commit_snapshot.c

Summary

Program for creating a new Base Image of the System Container from a snapshot
(Corresponds to "dre-sys commit -n <name> -t <tag> -b <base_image>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
10          char *snapshot = NULL;
11          char *base = NULL;
            . . . (snip) . . .
27          /* open handle */
28          h = dre_sys_open(name);
```

```
29        if(!h) {
30              result = DRE_ERR_NULL_POINTER;
31              goto error;
32        }
33
34        /* commit snapshot */
35        result = dre_sys_commit_snapshot(h, snapshot, base);
36        if(result) goto error;
37
38        /* close handle */
39        result = dre_sys_close(h);
40        if(result) goto error;
          . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_commit.c" for the full text of the source code.)

In Line 28, "dre_sys_open()" gets the handle of "<name>". In Line 35, "dre_sys_commit_snapshot()" creates a new Base Image as "<base_image>" from the snapshot named "<tag>" of the System Container named "<name>". In Line 39, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_commit_snapshot -n <name> -t <tag> -b <base_image>

Arguments

<name>	the name of the System Container
<tag>	the name of the snapshot
<base_image>	the name of the Base Image

1.2.2.4 dre_sys_create_cgroup.c

Summary

Program for creating a Control Group in the System Container
(Corresponds to "dre-sys create [-n <name>] -g <cgroup>")

Description

```
3   #include "dre_sys.h"
4
5   int main(int argc, char **argv)
6   {
7         dre_return_t result;
8         dre_sys_handle *h = NULL;
9         char *name = NULL;
10        char *cgroup = NULL;
          . . . (snip) . . .
24        /* open handle */
25        if(name) {
26              h = dre_sys_open(name);
27              if(!h) {
```

```
28                  result = DRE_ERR_NULL_POINTER;
29                  goto error;
30              }
31      }
32
33      /* create control group */
34      result = dre_sys_create_cgroup(h, cgroup);
35      if(result) goto error;
36
37      /* close handle */
38      if(h) {
39              result = dre_sys_close(h);
40              if(result) goto error;
41      }
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_create_cgroup.c" for the full text of the source code.)

In Line 26, "dre_sys_open()" gets the handle of "<name>", if the name of the System Container is specified as "<name>". In Line 34, "dre_sys_create_cgroup()" generates a Control Group named "<cgroup>". If "<name>" is specified, "<cgroup>" is generated on "<name>". Otherwise, "<cgroup>" is generated in the Host OS. In Line 39, "dre_sys_close()" releases the handle of "<name>", if "<name>" is specified.

Usage

dre_sys_create_cgroup -g <cgroup> [-n <name>]

Arguments

<cgroup>	the name of the Control Group
<name>	the name of the System Container (optional)

1.2.2.5 dre_sys_create_container.c

Summary

Program for creating a System Container
(Corresponds to "dre-sys create -n <name>")

Description

```
3   #include "dre_sys.h"
4
5   int main(int argc, char **argv)
6   {
7           dre_return_t result;
8           char *name = NULL;
            . . . (snip) . . .
20          /* create container */
21          dre_sys_data data = {DRE_SYS_CONTAINER_TYPE_KVM, 512, NULL, NULL};
22          result = dre_sys_create_container(name, &data);
23          if(result) goto error;
```

```
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_create_container.c" for the full text of the source code.)

In Line 21, the setting values of "<name>" are specified. In the above example, the setting values of "<name>" are the following.

- Use a Container of KVM
- Allocate 512 MB of memory
- Use the default Base Image
- Use the OS which is installed on the base image (Partition is not specified)

In Line 22, "dre_sys_create_container" creates the System Container using the values of "<name>" (specified by the arguments) and the setting specified In Line 15.

Usage

der_sys_create_container -n <name>

Arguments

<name>	the name of the System Container

1.2.2.6 dre_sys_destroy_cgroup.c

Summary

Program for removing a Control Group in the System Container
(Corresponds to "dre-sys destroy [-n <name>] -g <cgroup>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
10          char *cgroup = NULL;
            . . . (snip) . . .
24          /* open handle */
25          if(name) {
26                  h = dre_sys_open(name);
27                  if(!h) {
28                          result = DRE_ERR_NULL_POINTER;
29                          goto error;
30                  }
31          }
32
33          /* destroy control group */
34          result = dre_sys_destroy_cgroup(h, cgroup);
```

```
35        if(result) goto error;
36
37        /* close handle */
38        if(h) {
39              result = dre_sys_close(h);
40              if(result) goto error;
41        }
          . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_destroy_cgoup.c" for the full text of the source code.)

In Line 26, "dre_sys_open()" gets the handle of "<name>", if the name of the System Container is specified as "<name>". In Line 34, "dre_sys_destroy_cgroup()" removes a Control Group named "<cgroup>". If "<name>" is specified, "<cgroup>" in "<name>" is removed. Otherwise, "<cgroup>" in the Host OS is removed. In Line 39, "dre_sys_close()" releases the handle of "<name>", if "<name>" is specified.

Usage

dre_sys_destroy_container -g <cgroup> [-n <name>]

Arguments

<cgroup>	the name of the Control Group
<name>	the name of the System Container (optional)

1.2.2.7 dre_sys_destroy_container.c

Summary

Program for removing a System Container
(Corresponds to "dre-sys destroy -n <name>")

Description

```
3   #include "dre_sys.h"
4
5   int main(int argc, char **argv)
6   {
7         dre_return_t result;
8         char *name = NULL;
          . . . (snip) . . .
20        /* destroy container */
21        result = dre_sys_destroy_container(name);
22        if(result) goto error;{
          . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_destroy_container.c" for the full text of the source code.)

In Line 21, "dre_sys_detroy_container()" removes the System Container named "<name>".

Usage

dre_sys_destroy_container -n <name>

Arguments

<name> the name of the System Container

1.2.2.8 dre_sys_get_base_list.c

Summary

Program for listing the Base Images of the System Containers
(Corresponds to "dre-sys list --base")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_list_t list;
            . . . (snip) . . .
14          /* get base image list */
15          result = dre_sys_get_base_list(&list);
16          if(result) goto error;
17          if(list) {
18                  while(*(list + i)) {
19                          printf("%s¥n", *(list + i));
20                          i++;
21                  }
22          }
23
24          /* release base image list */
25          if(list) {
26                  result = dre_sys_release_base_list(list);
27                  if(result) goto error;
28          }
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_get_base_list.c" for the full text of the source code.)

In Line 15, "dre_sys_get_base_list()" gets the list of the Base Images in the system. In Lines 18 to 21, the names of the Base Images in the list are enumerated. In Line 26, "dre_sys_release_base_list()" releases the list of the name of the Base Images.

Usage

dre_sys_get_base_list

Arguments

None

1.2.2.9 dre_sys_get_checkpoint_list.c

Summary

Program for listing the checkpoints of a System Container
(Corresponds to "dre-sys list --checkpoint -n <name>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_app_handle *h = NULL;
 9          dre_list_t list;
10          char *name = NULL;
            . . . (snip) . . .
22          /* open handle */
23          h = dre_sys_open(name);
24          if(!h) {
25                  result = DRE_ERR_NULL_POINTER;
26                  goto error;
27          }
28
29          /* get checkpoint list */
30          result = dre_sys_get_checkpoint_list(h, &list);
31          if(result) goto error;
32          if(list) {
33                  int j = 0;
34                  while(*(list + j)) {
35                          printf("%s¥n", *(list + j));
36                          j++;
37                  }
38          }
39
40          /* release checkpoint list */
41          if(list) {
42                  result = dre_sys_release_checkpoint_list(list);
43                  if(result) goto error;
44          }
45
46          /* close handle */
47          result = dre_sys_close(h);
48          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_get_checkpoint_list.c" for the full text of the source code.)

In Line 23, "dre_sys_open()" gets the handle of "<name>". In Line 30, "dre_sys_get_checkpoint_list()" obtains a list of the checkpoints. In Lines 33 to 37, the names of the checkpoints in the list are enumerated. In Line 42, "dre_sys_release_checkpoint_list()" releases the list of the checkpoints. In Line 47, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_get_checkpoint_list -n <name>

Arguments

<name> the name of the System Container

1.2.2.10 dre_sys_get_container_list.c

Summary

Program for listing the System Containers on the system
(Corresponds to "dre-sys list")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_list_t list;
            . . . (snip) . . .
14          /* get container list */
15          result = dre_sys_get_container_list(&list);
16          if(result) goto error;
17          if(list) {
18                  while(*(list + i)) {
19                          printf("%s¥n", *(list + i));
20                          i++;
21                  }
22          }
23
24          /* release container list */
25          if(list) {
26                  result = dre_sys_release_container_list(list);
27                  if(result) goto error;
28          }
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_get_container_list.c" for the full text of the source code.)

In Line 15, "dre_sys_get_container_list()" obtains a list of the name of the System Containers on the system. In Lines 18 to 21, the names of the System Containers in the list are emumerated. In Line 26, "dre_sys_release_container_list()" releases the list of the name of the System Containers.

Usage

dre_sys_get_container_list

Arguments

None

1.2.2.11 dre_sys_get_resource.c

Summary

Program for obtaining the information of the resource usage of a System Container
(Corresponds to "dre-sys usage -n <name> -pid <pid> [-g <cgroup>]")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_sys_resource_value_t resource;
 8          dre_return_t result:
 9          dre_sys_handle *h = NULL;
10          char *name = NULL;
11          char *type = NULL;
12          char *cgroup = NULL;
13          char *app = NULL;
            . . . (snip) . . .
40          /* open handle */
41          h = dre_sys_open(name);
42          if(!h) {
43                  result = DRE_ERR_NULL_POINTER;
44                  goto error;
45          }
46
47          /* get resource usage */
48          if(!strcmp(type, "cpu")) {
49                  resource.type = DRE_SYS_CONTAINER_RESOURCE_CPU;
50                  result = dre_sys_get_resource(h, pid, &resource);
51                  if(result) goto error;
52                  printf("cpu usage:       %d%%¥n", resource.udata.cpu.percent);
53          } else if(!strcmp(type, "core")) {
54                  resource.type = DRE_SYS_CONTAINER_RESOURCE_CPU_CORE;
55                  result = dre_sys_get_resource(h, pid, &resource);
56                  if(result) goto error;
57                  printf("assined cpu core(s):       %s¥n", resource.udata.cpu.cores);
58          } else if(!strcmp(type, "mem")) {
59                  resource.type = DRE_SYS_CONTAINER_RESOURCE_MEM;
60                  result = dre_sys_get_resource(h, pid, &resource);
61                  if(result) goto error;
62                  printf("memory usage: %dBytes¥n", resource.udata.mem.usage_in_bytes);
63          } else if(!strcmp(type, "memsw")) {
64                  resource.type = DRE_SYS_CONTAINER_RESOURCE_MEMSW;
65                  result = dre_sys_get_resource(h, pid, &resource);
66                  if(result) goto error;
```

```
67              printf("memory+swap usage: %dBytes¥n", resource.udata.mem.memsw_usage_in_bytes);
68      } else {
69              printf("Invalid resource type: %s¥n", type);
70              result = DRE_ERR_INVALID_ARGUMENT;
71              goto error;
72      }
73
74      /* close handle */
75      result = dre_sys_close(h);
76      if(result) goto error;
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_get_resource.c" for the full text of the source code.)

In Line 41, "dre_sys_open()" gets the handle of "<name>". In Lines 48 to 72, "dre_sys_get_resource()" obtains the information of the resource usage and outputs it. "dre_sys_get_resource()" works as follows.

- It obtains the information of the resource usage of "<cgroup>" in "<application_container>" in "<name>", if "<cgroup>" and "<application_container>" are specified.
  - In addition to above, it obtains the information of the resource usage of "<pid>" which is bound to "<cgroup>", if "<pid>" is specified.
- It obtains the information of the resource usage of "<cgroup>" in "<name>", if "<application_container>" is omitted even though "<cgroup>" is specified.
  - In addition to above, it obtains the information of the resource usage of "<pid>" which is bound to "<cgroup>", if "<pid>" is specified.
- It obtains the information of the resource usage of "<application_container>" in "<name>", if "<cgroup>" is omitted even though "<application_container>" is specified.
  - In addition to above, it obtains the information of the resource usage of "<pid>" in "<application_container>", if "<pid>" is specified.
- It obtains the information of the resource usage of "<name>", if "<cgroup>" and "<application_container>" are omitted.
  - In addition to above, it obtains the information of the resource usage of "<pid>" in "<name>", if "<pid>" is specified.

In Line 75, "dre_sys_close()" releases the handle of "<name>".

Usage

```
dre_sys_get_resource   -n <name> <resource_type>  [--pid <pid>]  [-g <cgroup>]
                       [-a <application_container>]
```

Arguments

<name>	the name of the System Container
<resource_type>	type of the resource "cpu" : CPU usage (%) "core" : CPU core(s) "mem" : Memory usage (bytes) "memsw" : Memory and Swap usage (bytes)
<pid>	process ID (optional)
<cgroup>	the name of the Control Group (optional)
<application_container>	the name of the Application Container (optional)

1.2.2.12 dre_sys_get_snapshot_list.c

Summary

Program for listing the snapshots of a System Container
(Corresponds to "dre-sys list --snapshot -n <name>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          dre_list_t list;
10          char *name = NULL;
            . . . (snip) . . .
22          /* open handle */
23          h = dre_sys_open(name);
24          if(!h) {
25                  result = DRE_ERR_NULL_POINTER;
26                  goto error;
27          }
28
29          /* get snapshot list */
30          result = dre_sys_get_snapshot_list(h, &list);
31          if(result) goto error;
32          if(list) {
33                  int j = 0;
34                  while(*(list + j)) {
35                          printf("%s¥n", *(list + j));
36                          j++;
37                  }
38          }
39
40          /* release snapshot list */
41          if(list) {
42                  result = dre_sys_release_snapshot_list(list);
43                  if(result) goto error;
44          }
45
46          /* close handle */
47          result = dre_sys_close(h);
48          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_get_snapshot_list.c" for the full text of the source code.)

In Line 23, "dre_sys_open()" gets the handle of "<name>". In Line 30, "dre_sys_get_snapshot_list()" obtains a list of the names of the snapshots. In Lines 33 to 37, the names of the snapshots in the list are enumerated. In Line 42, "dre_sys_release_snapshot_list()" releases the list of the snapshots. In Line 47, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_get_snapshot_list -n <name>

Arguments

<name>	the name of the System Container

1.2.2.13 dre_sys_get_status.c

Summary

Program for getting the status of a System Container
(Corresponds to "dre-sys status -n <name>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
            . . . (snip) . . .
21          /* open handle */
22          h = dre_sys_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* get status */
29          result = dre_sys_get_status(h);
30          if(result == DRE_SYS_CONTAINER_STATUS_STOP) {
31                  printf("'%s' is STOPPED.¥n", name);
32          } else if(result == DRE_SYS_CONTAINER_STATUS_RUNNING) {
33                  printf("'%s' is RUNNING.¥n", name);
34          } else if(result == DRE_SYS_CONTAINER_STATUS_SUSPENDED) {
35                  printf("'%s' is SUSPENDED.¥n", name);
36          } else if(result == DRE_SYS_CONTAINER_STATUS_BUSY) {
37                  printf("'%s' is BUSY.¥n", name);
38          } else {
39                  result = DRE_ERR_CONTAINER_UNKNOWN_STATUS;
40                  goto error;
41          }
```

```
42
43        /* close handle */
44        result = dre_sys_close(h);
45        if(result) goto error;
          . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_get_status.c" for the full text of the source code.)

In Line 22, "dre_sys_open()" gets the handle of "<name>". In Line 29, "dre_sys_get_status()" gets the status of "<name>". In Lines 30 to 41, the result of "dre_sys_get_status()" is output. In Line 44, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_get_status -n <name>

Arguments

<name>	the name of the System Container

1.2.2.14 dre_sys_immigration_start.c

Summary

Program for starting a virtual machine of immigration in the target host in order to perform immigration of the System Container

(Corresponds to "dre-sys immigrate -n <name> -p <port>")

Description

```
3   #include "dre_sys.h"
4
5   int main(int argc, char **argv)
6   {
7         dre_return_t result:
8         int port;
          . . . (snip) . . .
23        /* start immigration */
24        result = dre_sys_immigration_start(h, port);
25        if(result) goto error;
          . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_immigration_start.c" for the full text of the source code.)

In Line 24, "dre_sys_immigration_start()" starts a virtual machine for the immigration of "<name>".

Usage

dre_sys_immigrate -n <name> -p <port>

Arguments

<name>	the name of the System Container
<port>	the port number for the migration

1.2.2.15 dre_sys_load_snapshot.c

Summary

Program for restoring the file system of a System Container
(Corresponds to "dre-sys load -n <name> -g <tag>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL
10          char *tag = NULL;
            . . . (snip) . . .
24          /* open handle */
25          h = dre_sys_open(name);
26          if(!h) {
27                  result = DRE_ERR_NULL_POINTER;
28                  goto error;
29          }
30
31          /* load snapshot */
32          result = dre_sys_load_snapshot(h, snapshot);
33          if(result) goto error;
34
35          /* close handle */
36          result = dre_sys_close(h);
37          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_load_snapshot.c" for the full text of the source code.)

In Line 25, "dre_sys_open()" gets the handle of "<name>". In Line 32, "dre_sys_load_snapshot()" restores the file system of the System Container named "<name>" with the snapshot named "<tag>". In Line 36, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_load_snapshot -n <name> -t <tag>

Arguments

<name>	the name of the System Container
<tag>	the name of the snapshot

1.2.2.16 dre_sys_migration_start.c

Summary

Program for performing the migration of the System Container
(Corresponds to "dre-sys migrate -n <name> -d <dest> -p <port>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          int port;
10          char *name = NULL;
11          char *dest = NULL;
            . . . (snip) . . .

27          /* open handle */
28          h = dre_sys_open(name);
29          if(!h) {
30                  result = DRE_ERR_NULL_POINTER;
31                  goto error;
32          }
33
34          /* start migration */
35          result = dre_sys_migration_start(h, dest, port);
36          if(result) goto error;
37
38          /* close handle */
39          result = dre_sys_close(h);
40          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_migration_start.c" for the full text of the source code.)

In Line 28, "dre_sys_open()" gets the handle of "<name>". In Line 35, "dre_sys_migration_start()" starts the migration of "<name>". In Line 39, "dre_sys_close()" releases the handle of "<name>". The "immigration" must be running at "<dest>" before the migration is performed.

Usage

dre_sys_migration_start -n <name> -d <dest> --port <port>

Arguments

<name>	the name of the System Container
<dest>	the host name or IP address of the destination of the migration
<port>	the port number for the migration

1.2.2.17 dre_sys_remove_base.c

Summary

Program for removing a Base Image of the System Containers
(Corresponds to "dre-sys remove --base <base_image>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h;
 9          char *base = NULL;
            . . . (snip) . . .
21          /* remove base image */
22          result = dre_sys_remove_base(base);
23          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_remove_base.c" for the full text of the source code.)

In Line 22, "dre_sys_remove_base()" removes the Base Image named "<name>".

Usage

dre_sys_remove_base -b <base_image>

Arguments

<base_image_name> the name of the Base Image

1.2.2.18 dre_sys_remove_checkpoint.c

Summary

Program for removing a checkpoint of a System Container
(Corresponds to "dre-sys remove --checkpoint -n <name> -t <tag>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
10          char *tag = NULL;
            . . . (snip) . . .
24          /* open handle */
25          h = dre_sys_open(name);
26          if(!h) {
27                  result = DRE_ERR_NULL_POINTER;
28                  goto error;
```

```
29          }
30
31          /* remove snapshot */
32          result = dre_sys_remove_checkpoint(h, snapshot);
33          if(result) goto error;
34
35          /* close handle */
36          result = dre_sys_close(h);
37          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_remove_checkpoint.c" for the full text of the source code.)

In Line 25, "dre_sys_open()" gets the handle of "<name>". In Line 32, "dre_sys_remove_checkpoint()" removes the checkpoint named "<tag>" of the System Container named "<name>". In Line 50, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_remove_checkpoint -n <name> -t <tag>

Arguments

<name>	the name of the System Container
<tag>	the name of the snapshot

1.2.2.19 dre_sys_remove_snapshot.c

Summary

Program for removing a snapshot of a System Container
(Corresponds to "dre-sys remove --snapshot -n <name> -t <tag>")

Description

```
3   #include "dre_sys.h"
4
5   int main(int argc, char **argv)
6   {
7           dre_return_t result:
8           dre_sys_handle *h = NULL;
9           char *name = NULL;
10          char *tag = NULL;
            . . . (snip) . . .
24          /* open handle */
25          h = dre_sys_open(name);
26          if(!h) {
27                  result = DRE_ERR_NULL_POINTER;
28                  goto error;
29          }
30
31          /* remove snapshot */
```

```
32        result = dre_sys_remove_snapshot(h, snapshot);
33        if(result) goto error;
34
35        /* close handle */
36        result = dre_sys_close(h);
37        if(result) goto error;
          . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_remove_snapshot.c" for the full text of the source code.)

In Line 25, "dre_sys_open()" gets the handle of "<name>". In Line 32, "dre_sys_remove_snapshot()" removes the snapshot named "<tag>" of the System Container named "<name>". In Line 50, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_remove_snapshot -n <name> -t <tag>

Arguments

<name>	the name of the System Container
<tag>	the name of the snapshot

1.2.2.20 dre_sys_restart.c

Summary

Program for restarting a System Container from a checkpoint
(Corresponds to "dre-sys restart -n <name> -t <tag>")

Description

```
3   #include "dre_sys.h"
4
5   int main(int argc, char **argv)
6   {
7         dre_return_t result:
8         dre_sys_handle *h = NULL;
9         char *name = NULL;
10        char *tag = NULL;
          . . . (snip) . . .
24        /* open handle */
25        h = dre_sys_open(name);
26        if(!h) {
27              result = DRE_ERR_NULL_POINTER;
28              goto error;
29        }
30
31        /* restart from checkpoint */
32        result = dre_sys_restart(h, tag);
33        if(result) goto error;
34
```

```
35        /* close handle */
36        result = dre_sys_close(h);
37        if(result) goto error;
          . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_restart.c" for the full text of the source code.)

In Line 25, "dre_sys_open()" gets the handle of "<name>". In Line 32, "dre_sys_restart()" restarts the System Container named "<name>" from the checkpoint named "<tag>". In Line 36, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_restart -n <name> -t <tag>

Arguments

<name>	the name of the System Container
<tag>	the name of the checkpoint

1.2.2.21 dre_sys_resume.c

Summary

Program for resuming a System Container which was suspended
(Corresponds to "dre-sys resume -n <name>")

Description

```
3   #include "dre_sys.h"
4
5   int main(int argc, char **argv)
6   {
7         dre_return_t result:
8         dre_sys_handle *h = NULL;
9         char *name = NULL;
          . . . (snip) . . .
21        /* open handle */
22        h = dre_sys_open(name);
23        if(!h) {
24              result = DRE_ERR_NULL_POINTER;
25              goto error;
26        }
27
28        /* resume container */
29        result = dre_sys_resume(h);
30        if(result) goto error;
31
32        /* close handle */
33        result = dre_sys_close(h);
34        if(result) goto error;
          . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_resume.c" for the full text of the source code.)

In Line 22, "dre_sys_open()" gets the handle of "<name>". In Line 29, "dre_sys_resume()" resumes the running of the System Container named "<name>" which was suspended. In Line 33, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_resume -n <name>

Arguments

<name>	the name of the System Container

1.2.2.22 dre_sys_save_snapshot.c

Summary

Program for generating a snapshot of a System Container
(Corresponds to "dre-sys save -n <name> -t <tag>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
10          char *tag = NULL;
            . . . (snip) . . .
24          /* open handle */
25          h = dre_sys_open(name);
26          if(!h) {
27                  result = DRE_ERR_NULL_POINTER;
28                  goto error;
29          }
30
31          /* save snapshot */
32          result = dre_sys_save_snapshot(h, snapshot);
33          if(result) goto error;
34
35          /* close handle */
36          result = dre_sys_close(h);
37          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_save_snapshot.c" for the full text of the source code.)

In Line 25, "dre_sys_open()" gets the handle of "<name>". In Line 32, "dre_sys_save_snapshot()" generates the snapshot of the file system of the System Container named "<name>" as "<tag>". In Line 36, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_save_snapshot -n <name> -t <tag>

Arguments

<name>	the name of the System Container
<tag>	the name of the snapshot

1.2.2.23 dre_sys_set_resource.c

Summary

Program for setting a limitation on resource usage of a System Container
(Corresponds to "dre-sys limit <resource_type> <limitation> -n <name> [-g <cgroup>] [-a <application_container>]")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_sys_resource_value_t resource;
 8          dre_return_t result:
 9          dre_sys_handle *h = NULL;
10          char *name = NULL;
11          char *cgroup = NULL;
12          char *type = NULL;
13          char *app = NULL;
14          char *limitation = NULL;
            . . . (snip) . . .
42          /* set resource usage */
43          if(!strcmp(type, "cpu")) {
44                  resource.type = DRE_SYS_CONTAINER_RESOURCE_CPU;
45                  resource.udata.cpu.percent = atoi(limitation);
46          } else if(!strcmp(type, "core")) {
47                  resource.type = DRE_SYS_CONTAINER_RESOURCE_CPU_CORE;
48                  resource.udata.cpu.cores = limitation;
49          } else if(!strcmp(type, "mem")) {
50                  resource.type = DRE_SYS_CONTAINER_RESOURCE_MEM;
51                  resource.udata.mem.usage_in_bytes = atoi(limitation);
52          } else if(!strcmp(type, "memsw")) {
53                  resource.type = DRE_SYS_CONTAINER_RESOURCE_MEMSW;
54                  resource.udata.mem.memsw_usage_in_bytes = atoi(limitation);
55          } else {
56                  printf("Invalid resource type: %s¥n", type);
57                  result = DRE_ERR_INVALID_ARGUMENT;
58                  goto error;
59          }
60
```

```
61      /* open handle */
62      if(name) {
63          h = dre_sys_open(name);
64          if(!h) {
65              result = DRE_ERR_NULL_POINTER;
66              goto error;
67          }
68      }
69
70      /* set the limitation */
71      result = dre_sys_set_resource(h, cgroup, app, resource);
72      if(result) goto error;
73      printf("Set the limitation of %s.¥n", type);
74
75      /* close handle */
76      if(h) {
77          result = dre_sys_close(h);
78          if(result) goto error;
79      }
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_set_resource.c" for the full text of the source code.)

In Lines 43 to 59 specify the resources whose use is to be limited. These lines work as follows.

- They set the limitation on the resource usage of "<cgroup>" in "<application_container>" in "<name>", if "<cgroup>" and "<application_container>" are specified.
- They set the limitation on the resource usage of "<cgroup>" in "<name>", if "<application_container>" is omitted even though "<cgroup>" is specified.
- They set the limitation on the resource usage of "<application_container>" in "<name>", if "<cgroup>" is omitted even though "<application_container>" is specified.
- They set the limitation on the resource usage of "<name>", if "<cgroup>" and "<application_container>" are omitted.

In Line 63, "dre_sys_open()" gets the handle of "<name>". In Line 71, "dre_sys_set_resource()" sets the limitation of resource usage based on the result of the lines from 43 to 59. In Line 77, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_set_resource -n <name> <resource_type> <limitation> [-g <cgroup>]
[-a <application_container>]

Arguments

<name>	the name of the System Container
<resource_type>	type of the resource
	"cpu" : CPU usage (%)
	"core" : CPU core(s)
	"mem" : Memory usage (bytes)
	"memsw" : Memory and Swap usage (bytes)
<limitation>	the limitation of the resource usage

<cgroup>	the name of the Control Group (optional)
<application_container>	the name of the Application Container (optional)

1.2.2.24 dre_sys_share.c

Summary

Program for sharing the System Container with an NFS server
(Corresponds to "dre-sys share -n <name>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
            . . . (snip) . . .
21          /* open handle */
22          h = dre_sys_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* share container */
29          result = dre_sys_share(h);
30          if(result) goto error;
31
32          /* close handle */
33          result = dre_sys_close(h);
34          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_share.c" for the full text of the source code.)

In Line 22, "dre_sys_open()" gets the handle of "<name>". In Line 29, "dre_sys_share()" shares the System Container named "<name>" with the NFS server. The NFS server that will be used for sharing must be mounted to ${DRE_NFS_PATH} in advance. (For ${DRE_NFS_PATH}, see "/etc/default/dre".) In Line 33, "dre_sys_close()" releases the handle of "<name>"

Usage

dre_sys_share -n <name>

Arguments

<name>	the name of the System Container

1.2.2.25 dre_sys_start.c

Summary

Program for starting a System Container
(Corresponds to "dre-sys start -n <name>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
            . . . (snip) . . .
21          /* open handle */
22          h = dre_sys_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* start container */
29          result = dre_sys_start(h);
30          if(result) goto error;
31
32          /* close handle */
33          result = dre_sys_close(h);
34          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_start.c" for the full text of the source code.)

In Line 22, "dre_sys_open()" gets the handle of "<name>". In Line 29, "dre_sys_start()" starts the System Container named "<name>". In Line 33, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_start -n <name>

Arguments

<name>	the name of the System Container

1.2.2.26 dre_sys_stop.c

Summary

Program for stopping a System Container
(Corresponds to "dre-sys stop -n <name>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
            . . . (snip) . . .
21          /* open handle */
22          h = dre_sys_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* stop container */
29          result = dre_sys_stop(h);
30          if(result) goto error;
31
32          /* close handle */
33          result = dre_sys_close(h);
34          if(result) goto error;
            . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_stop.c" for the full text of the source code.)

In Line 22, "dre_sys_open()" gets the handle of "<name>". In Line 29, "dre_sys_stop()" stops the System Container named "<name>" which is running. In Line 33, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_stop -n <name>

Arguments

<name>	the name of the System Container

1.2.2.27 dre_sys_suspend.c

Summary

Program for suspending a System Container which is running
(Corresponds to "dre-sys suspend -n <name>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
```

```
 8        dre_sys_handle *h = NULL;
 9        char *name = NULL;
          . . . (snip) . . .
21        /* open handle */
22        h = dre_sys_open(name);
23        if(!h) {
24              result = DRE_ERR_NULL_POINTER;
25              goto error;
26        }
27
28        /* suspend container */
29        result = dre_sys_suspend(h);
30        if(result) goto error;
31
32        /* close handle */
33        result = dre_sys_close(h);
34        if(result) goto error;
          . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_suspend.c" for the full text of the source code.)

In Line 22, "dre_sys_open()" gets the handle of "<name>". In Line 29, "dre_sys_suspend()" suspends the System Container named "<name>" which is running. In Line 33, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_suspend -n <name>

Arguments

<name>	the name of the System Container

1.2.2.28 dre_sys_unbind_pid.c

Summary

Program for unbinding a process which is bound to the Control Group on a System Container (Corresponds to "dre-sys unbind [-n <name>] -g <cgroup> -pid <pid>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7        dre_return_t result:
 8        dre_sys_handle *h = NULL;
 9        char *name = NULL;
10        char *cgroup = NULL;
          . . . (snip) . . .
27        /* open handle */
28        if(name) {
```

```
29              h = dre_sys_open(name);
30              if(!h) {
31                      result = DRE_ERR_NULL_POINTER;
32                      goto error;
33              }
34      }
35
36      /* unbind from control group */
37      result = dre_sys_unbind_pid(h, cgroup, pid);
38      if(result) goto error;
39
40      /* close handle */
41      if(h) {
42              result = dre_sys_close(h);
43              if(result) goto error;
44      }
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_unbind_pid.c" for the full text of the source code.)

In Line 29, "dre_sys_open()" gets the handle of "<name>", if the name of the System Container is specified as "<name>". In Line 37, "dre_sys_unbind_pid()" unbinds "<pid>" from the Control Group specified as "<cgroup>". At this time, "<cgroup>" means the Control Group in the "<name>" if "<name>" is specified. If "<name>" is not specified, "<cgroup>" means the Control Group in the Host OS. In Line 42, "dre_sys_close()" releases the handle of "<name>", if "<name>" is specified.

Usage

dre_sys_unbind_pid -g <cgroup> --pid<pid> [-n <name>]

Arguments

<cgroup>	the name of the Control Group
<pid>	process ID
<name>	the name of the System Container

1.2.2.29 dre_sys_unset_resource.c

Summary

Program for releasing the limitation on resource usage of a System Container
(Corresponds to "dre-sys unlimit <resource_type> -n <name> [-g <cgroup>] [-a <application_container>]")

Description

```
3   #include "dre_sys.h"
4
5   int main(int argc, char **argv)
6   {
7           dre_sys_resource_value_t resource;
8           dre_return_t result;
9           dre_sys_handle *h = NULL;
```

```
10      char *name = NULL;
11      char *cgroup = NULL;
12      char *type = NULL;
13      char *app = NULL;
14      char *limitation = NULL;
        . . . (snip) . . .
37      /* unset resource usage */
38      if(!strcmp(type, "cpu")) {
39          resource.type = DRE_SYS_CONTAINER_RESOURCE_CPU;
40      } else if(!strcmp(type, "core")) {
41          resource.type = DRE_SYS_CONTAINER_RESOURCE_CPU_CORE;
42      } else if(!strcmp(type, "mem")) {
43          resource.type = DRE_SYS_CONTAINER_RESOURCE_MEM;
44      } else if(!strcmp(type, "memsw")) {
45          resource.type = DRE_SYS_CONTAINER_RESOURCE_MEMSW;
46      } else {
47          printf("Invalid resource type: %s¥n", type);
48          result = DRE_ERR_INVALID_ARGUMENT;
49          goto error;
50      }
51
52      /* open handle */
53      if(name) {
54          h = dre_sys_open(name);
55          if(!h) {
56              result = DRE_ERR_NULL_POINTER;
57              goto error;
58          }
59      }
60
61      /* unset the limitation */
62      result = dre_sys_unset_resource(h, cgroup, app, resource);
63      if(result) goto error;
64      printf("Unset the limitation of %s.¥n", type);
65
66      /* close handle */
67      if(h) {
68          result = dre_sys_close(h);
69          if(result) goto error;
70      }
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_unset_resource.c" for the full text of the source code.)

In Lines 38 to 50 specify the resource whose usage limitation is to be released. These lines work as follows.

- They release the limitation on the resource usage of "<cgroup>" in "<application_container>" in "<name>", if "<cgroup>" and "<application_container>" are specified.
- They release the limitation on the resource usage of "<cgroup>" in "<name>", if "<application_container>" is omitted even though "<cgroup>" is specified.

- They release the limitation on the resource usage of "<application_container>" in "<name>", if "<cgroup>" is omitted even though "<application_container>" is specified.
- They release the limitation on the resource usage of "<name>", if "<cgroup>" and "<application_container>" are omitted.

In Line 54, "dre_sys_open()" gets the handle of "<name>". In Line 62, "dre_sys_unset_resource()" unsets the limitation on resource usage based on the result of the lines from 38 to 50. In Line 68, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_unset_resource -n<name> <resource_type> [-g <cgroup>] [-a <application_container>]

Arguments

<name>	the name of the System Container
<resource_type>	type of the resource
	"cpu" : CPU usage (%)
	"core" : CPU core(s)
	"mem" : Memory usage (bytes)
	"memsw" : Memory and Swap usage (bytes)
<cgroup>	the name of the Control Group (optional)
<application_container>	the name of the Application Container (optional)

1.2.2.30 dre_sys_unshare.c

Summary

Program for stopping sharing the System Container between the Host OS and the NFS server (Corresponds to "dre-sys unshare -n <name>")

Description

```
 3  #include "dre_sys.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
            . . . (snip) . . .
21          /* open handle */
22          h = dre_sys_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* share container */
29          result = dre_sys_unshare(h);
```

```
30      if(result) goto error;
31
32      /* close handle */
33      result = dre_sys_close(h);
34      if(result) goto error;
        . . . (snip) . . .
```

(See "/usr/share/doc/libdre0-dev/sample/sys/dre_sys_unshare.c" for the full text of the source code.)

In Line 22, "dre_sys_open()" gets the handle of "<name>". In Line 29, "dre_sys_unshare()" unshares the System Container named "<name>" with the NFS server. In Line 33, "dre_sys_close()" releases the handle of "<name>".

Usage

dre_sys_unshare -n <name>

Arguments

<name>	the name of the System Container

2 Sample programs which call the APIs of the D-Aware Applications

2.1 How to use

When libdaware0-dev is installed, the source codes for the sample programs that use these APIs will be installed in the following path.

/usr/share/doc/libdaware0-dev/sample/

Run "make" in the above path to compile the samples.

2.2 Description of the sample programs

The following samples are available.

daware-termination.c

Program for registering/unregistering the end callback function

daware-log.c

Program for outputting a formatted log to the syslog

2.2.1 daware-termination.c

Summary

Program for registering/unregistering the end callback function

The end callback will start when it receives a signal. The termination process is called by the end callback and terminates the program.

Description

```
 3  #include "daware.h"
    . . . (snip) . . .
 7  int termination_flag = 0;
 8
 9  int
10  termination_callback (int signo, void *arg)
11  {
12          termination_flag = 1;
13          return 0;
14  }
15
16  int
17  main (int argc, char *argv[])
18  {
            . . . (snip) . . .
21          DTermination *dterm = daware_termination_register_callback
22          (SIGTERM, termination_callback, NULL);
            . . . (snip) . . .
33          while (1) {
```

```
34              printf("%d¥n", val);
35              if (termination_flag) break;
36                      val++;
37              sleep(1);
38      }
39
40      fp = fopen(DATA_FILE, "wb");
41      if (fp) {
42              printf("save: %d > %s¥n", val, DATA_FILE);
43              fwrite(&val, sizeof(val), 1, fp);
44              fclose(fp);
45      }
46
47      daware_termination_unregister_callback(dterm);
        . . . (snip) . . .
```

(See "/usr/share/doc/libdaware0-dev/sample/daware-termination.c" for the full text of the source code.)

In Line 21, "daware_termination_register_callback()" registers the end callback function. In this example, "termination_callback()" is started when the SIGTERM is received. After receiving the SIGTERM, the "termination_callback()" In Line 10 sets "termination_flag" as "1". "termination_flag" is monitored at the loop In Line from 33 to 38. When "termination_flag" is turned to "1", this loop is exited and the termination process In Line from 40 to 45 will be called. In Line 47, "daware_termination_unregister_callback()" unregisters the end callback function.

Usage

Run "daware-termination" to kill it from another terminal.

```
$ ./daware-termination
PID: 7217
0
1
2
3
save: 4 > /tmp/daware-termination.dat
```

```
$ kill 7217
```

In the above, "kill" is called when "3" has been displayed and the termination process which writes data into "/tmp/daware-termination.dat" is called.

2.2.2daware-log.c

Summary

Program for outputting a formatted log to the syslog

Description

```
1   #include <stdio.h>
```

```
 2  #include "daware.h"
 3
 4  int main (int argc, char *argv[])
 5  {
 6          const char format[] = "%faci%.%lvl% %file% line %line%: %msg%";
 7
 8          daware_log_open(argv[0], LOG_CONS|LOG_PID, LOG_LOCAL7);
 9
10          daware_log_set_format(format);
11
12          DLOG_NOTICE("message 1");
13          DLOG_INFO("message 2");
14          DLOG_DEBUG("message 3");
15
16          DLOG(LOG_USER|LOG_DEBUG, "message 4");
17
18          daware_log_close();
19
20          return 0;
21  }
```

(See "/usr/share/doc/libdaware0-dev/sample/daware-log.c" for the full text of the source code.)

In Line 8, the options and the facility of "syslog" are set and communication with "syslog" is established. In Line 6 and 10, the format of the log is set. The facility and level of syslog in the places "%faci%" and "%lvl%", the line number in the place "%line%", and the message in the place "%msg%" are written into the syslog. In Lines 12 to 14, each macro which matches the level of the syslog is called. The macro named "DLOG" In Line 16 sets the value of the facility and the level. In the above sample, the facility is set as LOG_USER and the level is set as LOG_DEBUG. In Line 18, "daware_log_close()" breaks the communication with syslog.

Usage

Run "daware-log" and check the output written in syslog.

```
$ ./daware-log
$ tail /var/log/syslog
Apr 17 16:36:55 ubuntu-12 ./daware-log[6977]: local5.notice daware-log.c line 12: message 1
Apr 17 16:36:55 ubuntu-12 ./daware-log[6977]: local5.info daware-log.c line 13: message 2
Apr 17 16:36:55 ubuntu-12 ./daware-log[6977]: local5.debug daware-log.c line 14: message 3
Apr 17 16:36:55 ubuntu-12 ./daware-log[6977]: user.debug daware-log.c line 16: message 4
```