NAME

org.freedesktop.portable1 − The D−Bus interface of systemd−portabled

INTRODUCTION

systemd-portabled.service(8) is a system service that may be used to attach, detach and inspect portable services. This page describes the D−Bus interface.

THE MANAGER OBJECT

The service exposes the following interfaces on the Manager object on the bus:

```
node /org/freedesktop/portable1 {
 interface org.freedesktop.portable1.Manager {
  methods:
   GetImage(in  s image,
            out o object);
   ListImages(out a(ssbtttso) images);
   GetImageOSRelease(in  s image,
                     out a{ss} os_release);
   GetImageMetadata(in  s image,
                    in  as matches,
                    out s image,
                    out ay os_release,
                    out a{say} units);
   GetImageMetadataWithExtensions(in  s image,
                                  in  as extensions,
                                  in  as matches,
                                  in  t flags,
                                  out s image,
                                  out ay os_release,
                                  out a{say} extensions,
                                  out a{say} units);
   GetImageState(in  s image,
                 out s state);
   AttachImage(in  s image,
               in  as matches,
               in  s profile,
               in  b runtime,
               in  s copy_mode,
               out a(sss) changes);
   AttachImageWithExtensions(in  s image,
                             in  as extensions,
                             in  as matches,
                             in  s profile,
                             in  s copy_mode,
                             in  t flags,
                             out a(sss) changes);
   DetachImage(in  s image,
               in  b runtime,
               out a(sss) changes);
   DetachImageWithExtensions(in  s image,
                             in  as extensions,
                             in  t flags,
                             out a(sss) changes);
   ReattachImage(in  s image,
                 in  as matches,
                 in  s profile,
```

```
                in  b runtime,
                in  s copy_mode,
                out a(sss) changes_removed,
                out a(sss) changes_updated);
    ReattachImageWithExtensions(in  s image,
                                in  as extensions,
                                in  as matches,
                                in  s profile,
                                in  s copy_mode,
                                in  t flags,
                                out a(sss) changes_removed,
                                out a(sss) changes_updated);
    RemoveImage(in  s image);
    MarkImageReadOnly(in  s image,
                      in  b read_only);
    SetImageLimit(in  s image,
                  in  t limit);
    SetPoolLimit(in  t limit);
  properties:
    @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
    readonly s PoolPath = '...';
    @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
    readonly t PoolUsage = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
    readonly t PoolLimit = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
    readonly as Profiles = ['...', ...];
 };
 interface org.freedesktop.DBus.Peer { ... };
 interface org.freedesktop.DBus.Introspectable { ... };
 interface org.freedesktop.DBus.Properties { ... };
};
```

Methods

GetImage() may be used to get the image object path of the image with the specified name.

ListImages() returns an array of all currently known images. The structures in the array consist of the following fields: image name, type, read−only flag, creation time, modification time, current disk space, usage and image object path.

GetImageOSRelease() retrieves the OS release information of an image. This method returns an array of key value pairs read from the **os-release**(5) file in the image and is useful to identify the operating system used in a portable service.

GetImageMetadata() retrieves metadata associated with an image. This method returns the image name, the image's **os-release**(5) content in the form of a (streamable) array of bytes, and a list of portable units contained in the image, in the form of a string (unit name) and an array of bytes with the content.

GetImageMetadataWithExtensions() retrieves metadata associated with an image. This method is a superset of **GetImageMetadata()** with the addition of a list of extensions as input parameter, which were overlaid on top of the main image via **AttachImageWithExtensions()**. The path of each extension and an array of bytes with the content of the respective extension−release file are returned, one such structure for each extension named in the input arguments.

GetImageState() retrieves the image state as one of the following strings:

- detached
- attached
- attached−runtime
- enabled
- enabled−runtime
- running
- running−runtime

AttachImage() attaches a portable image to the system. This method takes an image path or name, a list of strings that will be used to search for unit files inside the image (partial or complete matches), a string indicating which portable profile to use for the image (see *Profiles* property for a list of available profiles), a boolean indicating whether to attach the image only for the current boot session, and a string representing the preferred copy mode (whether to copy the image or to just symlink it) with the following possible values:

- (null)
- copy
- symlink

This method returns the list of changes applied to the system (for example, which unit was added and is now available as a system service). Each change is represented as a triplet of strings: the type of change applied, the path on which it was applied, and the source (if any). The type of change applied will be one of the following possible values:

- copy
- symlink
- write
- mkdir

Note that an image cannot be attached if a unit that it contains is already present on the system.

AttachImageWithExtensions() attaches a portable image to the system. This method is a superset of **AttachImage()** with the addition of a list of extensions as input parameter, which will be overlaid on top of

the main image. When this method is used, detaching must be done by passing the same arguments via the **DetachImageWithExtensions()** method. For more details on this functionality, see the *MountImages=* entry on **systemd.exec**(5) and **systemd-sysext**(8). The *flag* parameter is currently unused and reserved for future purposes.

DetachImage() detaches a portable image from the system. This method takes an image path or name, and a boolean indicating whether the image to detach was attached only for the current boot session or persistently. This method returns the list of changes applied to the system (for example, which unit was removed and is no longer available as a system service). Each change is represented as a triplet of strings: the type of change applied, the path on which it was applied, and the source (if any). The type of change applied will be one of the following possible values:

- unlink

Note that an image cannot be detached if a unit that it contains is running.

DetachImageWithExtensions() detaches a portable image from the system. This method is a superset of **DetachImage()** with the addition of a list of extensions as input parameter, which were overlaid on top of the main image via **AttachImageWithExtensions()**. The *flag* parameter is currently unused and reserved for future purposes.

ReattachImage() combines the effects of the **AttachImage()** method and the **DetachImage()** method. The difference is that it is allowed to reattach an image while one or more of its units are running. The reattach operation will fail if no matching image is attached. The input parameters match the **AttachImage()** method, and the return parameters are the combination of the return parameters of the **DetachImage()** method (first array, units that were removed) and the **AttachImage()** method (second array, units that were updated or added).

ReattachImageWithExtensions() reattaches a portable image to the system. This method is a superset of **ReattachImage()** with the addition of a list of extensions as input parameter, which will be overlaid on top of the main image. For more details on this functionality, see the *MountImages=* entry on **systemd.exec**(5) and **systemd-sysext**(8). The *flag* parameter is currently unused and reserved for future purposes

RemoveImage() removes the image with the specified name.

MarkImageReadOnly() toggles the read−only flag of an image.

SetPoolLimit() sets an overall quota limit on the pool of images.

SetImageLimit() sets a per−image quota limit.

The **AttachImageWithExtensions()**, **DetachImageWithExtensions()** and **ReattachImageWithExtensions()** methods take in options as flags instead of booleans to allow for extendability, defined as follows:

```
#define SD_SYSTEMD_PORTABLE_RUNTIME  (UINT64_C(1) << 0)
```

Properties

PoolPath specifies the file system path where images are written to.

PoolUsage specifies the current usage size of the image pool in bytes.

PoolLimit specifies the size limit of the image pool in bytes.

Profiles specifies the available runtime profiles for portable services.

THE IMAGE OBJECT

The service exposes the following interfaces on the Image object on the bus:

```
node /org/freedesktop/portable1 {
  interface org.freedesktop.portable1.Image {
    methods:
```

```
  GetOSRelease(out a{ss} os_release);
  GetMetadata(in  as matches,
              out s image,
              out ay os_release,
              out a{say} units);
  GetMetadataWithExtensions(in  as extensions,
                            in  as matches,
                            in  t flags,
                            out s image,
                            out ay os_release,
                            out a{say} extensions,
                            out a{say} units);
  GetState(out s state);
  Attach(in  as matches,
         in  s profile,
         in  b runtime,
         in  s copy_mode,
         out a(sss) changes);
  AttachWithExtensions(in  as extensions,
                       in  as matches,
                       in  s profile,
                       in  s copy_mode,
                       in  t flags,
                       out a(sss) changes);
  Detach(in  b runtime,
         out a(sss) changes);
  DetachWithExtensions(in  as extensions,
                       in  t flags,
                       out a(sss) changes);
  Reattach(in  as matches,
           in  s profile,
           in  b runtime,
           in  s copy_mode,
           out a(sss) changes_removed,
           out a(sss) changes_updated);
  ReattacheWithExtensions(in  as extensions,
                          in  as matches,
                          in  s profile,
                          in  s copy_mode,
                          in  t flags,
                          out a(sss) changes_removed,
                          out a(sss) changes_updated);
  Remove();
  MarkReadOnly(in  b read_only);
  SetLimit(in  t limit);
 properties:
  @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
  readonly s Name = '...';
  @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
  readonly s Path = '...';
  @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
  readonly s Type = '...';
  @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
  readonly b ReadOnly = ...;
```

```
    @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
    readonly t CreationTimestamp = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
    readonly t ModificationTimestamp = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
    readonly t Usage = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
    readonly t Limit = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
    readonly t UsageExclusive = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
    readonly t LimitExclusive = ...;
  };
  interface org.freedesktop.DBus.Peer { ... };
  interface org.freedesktop.DBus.Introspectable { ... };
  interface org.freedesktop.DBus.Properties { ... };
};
```

Methods

The following methods implement the same operation as the respective methods on the Manager object (see above). However, these methods operate on the image object and hence does not take an image name parameter. Invoking the methods directly on the Manager object has the advantage of not requiring a **GetImage()** call to get the image object for a specific image name. Calling the methods on the Manager object is hence a round trip optimization. List of methods:

- GetOSRelease()
- GetMetadata()
- GetMetadataWithExtensions()

- GetState()
- Attach()
- AttachWithExtensions()
- Detach()
- DetachWithExtensions()
- Reattach()
- ReattacheWithExtensions()
- Remove()
- MarkReadOnly()
- SetLimit()

Properties

Name specifies the image name.

Path specifies the file system path where image is stored.

Type specifies the image type.

ReadOnly specifies whether the image is read−only.

CreationTimestamp specifies the image creation timestamp.

ModificationTimestamp specifies the image modification timestamp.

Usage specifies the image disk usage.

Limit specifies the image disk usage limit.

UsageExclusive specifies the image disk usage (exclusive).

LimitExclusive specifies the image disk usage limit (exclusive).

VERSIONING

These D−Bus interfaces follow **the usual interface versioning guidelines**[1].

NOTES

1. the usual interface versioning guidelines
   http://0pointer.de/blog/projects/versioning-dbus.html