Stratisd D-Bus API Reference Manual*
Stratisd Version 0.0.1

Anne Mulhern

Version 0.1
Last modified: 03/06/2017

Contents

Asking Questions and Making Changes to this Document

This document can be foundin the stratis-docs repo, and is written using LyX 2.2.2. Please ask any questions by opening an issue, and propose changes as pull requests.

1 Introduction

This document describes the D-Bus API for the Stratis daemon. The D-Bus API constitutes the only public API of the Stratis daemon at the time of this writing. The public methods of the daemon's engine do not constitute part of the public API. The D-Bus API is a thin layer which receives message on the

D-Bus, processes them, transmits them to the Stratis engine, receives the results from the engine, and transmits a response on the D-Bus.

1.1 Technical Notes

1.1.1 Constants

The top level D-Bus object exports lists of certain constant values that it uses via properties. This allows the client to identify the constants by their name rather than by their number and be robust to changes in ordering or the addition of new constants by the daemon.

1.1.2 Emulating an Option type

The D-Bus specification reserves a signature code for a "maybe" or "option" type, but this type is not available[1]. Nonetheless, it is desirable to have a way of representing a "don't care" condition when invoking D-Bus methods. Certain methods accept a pair argument that mimics such a type. The first item is a boolean. If the value is false, then the pair represents None. If the value is true, then the pair represents Some(x) where x is the second item of the pair.

2 Standard Interfaces and Methods

2.1 org.freedesktop.dbus.ObjectManager interface

The top level D-Bus object implements the org.freedesktop.dbus.ObjectManager interface. This interface defines the GetManagedObjects() method, which returns a view of the objects that the D-Bus layer has in its tree. This view constitutes a summary of the state of the pools, devices, and filesystem which Stratis manages. The objects are identified primarily by their D-Bus object paths. However, depending on the interface which an object supports it may also support certain identifying properties. For example, all objects that represent pools support an interface which implements a Name and a Uuid property which may be used by the client to identify a pool.

In invoking methods, or obtaining the values of properties, the client must identify existing objects by means of their object paths, either implicitly by invoking a method on an object constructed from an object path, or explicitly, by passing object paths as arguments to a method. For example, the API's DestroyPool() method requires two object paths:

- the object path of the Manager object, on which to invoke the DestroyPool method
- the object path of the pool object, which is passed as an argument to the DestroyPool method

[1] `https://dbus.freedesktop.org/doc/dbus-specification.html`

It is expected that the client will identify the object path of the pool object by locating it using its Name or Uuid property.

2.2 org.freedesktop.dbus.Introspectable

All objects support the org.freedesktop.dbus.Introspectable interface. This interface has one method, Introspect(), which returns an XML description of the object's interfaces, methods, and properties.

3 Stratisd Interfaces and Methods

Each kind of object implements an identifying interface, i.e., pools implement a pool interface, filesystems implement a filesystem interface, etc. There is a top level manager interface that implements management tasks. The following is a description of the interfaces and their methods and properties. Some more detailed information can be obtained from the introspection information that can be queried for each object, including the signatures of all methods and properties. All methods return a return code and an accompanying message. Some methods also return a result. If a method does return a result, then the result is the first element in a triple, otherwise the returned value is just a pair of the return code and the message.

3.1 Manager interface

The Manager interface is the top level interface. It manages the creation and destruction of pools and also exports various global properties.

Methods

ConfigureSimulator This method is solely used to configure the simulator engine. Invoking it on the real engine is a no-op.

Arguments:

denominator the denominator of the fraction that determines the frequency of unusual occurences, generally failures. Signature: u.

Returns: nothing

CreatePool This method creates a single pool with the specified name and blockdevs.

Arguments:

name the name of the pool. Signature: s.

redundancy the redundancy specification for the pool. Signature: (bq). Note that redundancy is an Option argument.

force if true, the method will claim all devices specified, regardless of whether they seem to belong to another application. Signature: b.

devices device nodes of devices to form the pool. Signature: as.

Returns: the object path of the constructed pool and the device nodes of the blockdevs added to the pool. Signature: (oas)qs.

DestroyPool This method destroys the specified pool.

Arguments:

pool_object_path the object path of the pool to destroy. Signature: o.

Returns: true if it was necessary to take an action, otherwise false. If the pool has already been destroyed, no action is required. Signature: bqs.

Properties

ErrorValues These are the values that are used as return codes by all the methods. This property is an array of pairs of names and corresponding numeric values. Signature: a(sq).

RedundancyValues This property specifies the redundancy values that stratisd supports. It has the same format as the ErrorValues property. Signature: a(sq).

Version The current stratisd version. Signature: s.

3.2 pool interface

The pool interface manages the devices and filesystems within a pool.

Methods

AddDevs This method allows additional devices to be added to the pool after it is created.

Arguments:

force this option has the same meaning as the force option of CreatePool. Signature: b.

devices this argument has the same meaning as the device argument of CreatePool. Signature: as.

Returns: the device nodes of the blockdevs added to the pool. Signature: asqs.

CreateFilesystems This method allows creating multiple filesystems, specified by name, on a pool.

Arguments:

specs The specification for each filesystem to be created, currently just a name. Signature: as.

Returns a list of pairs of object paths and names of filesystems created. Signature: a(os)qs.

DestroyFilesystems This method allows destroying multiple filesystems.

Arguments:

filesystems the object paths of the filesystems to be destroyed. Signature: ao.

Returns: the names of the filesystems destroyed. Signature: asqs.

SetName This allows setting the name of the pool.

Arguments:

new_name The name to set. Signature: s.

Returns: true if the name was actually changed, otherwise false. Signature: bqs.

Properties

Name The name of the pool. Signature: s.

TotalPhysicalSize The total physical size of the pool in sectors. These sectors may be used by the pool for many purposes, so there are always fewer sectors than this that can be used to store user data. This is the largest physical size that can be meaningfully associated with a pool. Signature: s.

TotalPhysicalUsed All the sectors in use by the pool for any purpose whatsoever. These purposes include storage of user data as well as recording of pool metadata and other pool administration. This value may never exceed TotalPhysicalSize. Signature: s.

Uuid The UUID of the pool. This property is constant. Signature: s.

3.3 filesystem interface

The fileystem interface manages the properties of individual filesystems.

Methods

CreateSnapshot This method allows a snapshot to be created from an existing filesystem.

Arguments:

name The name of the snapshot. Signature: s.

Results: the object path of the newly created filesystem. Signature: oqs.

SetName This allows setting the name of the filesystem.

Arguments:

name The name to set. Signature: s.

Returns: true if the name was actually changed, otherwise false. Signature: bqs.

Properties

Devnode The device node of the filesystem's corresponding thin device. This property is constant. Signature: s.

Name The name of the filesystem. Signature: s.

Pool The object path of the parent pool. This property is constant. Signature: o.

Uuid The UUID of the filesystem. This property is constant. Signature: s.