Stratisd D-Bus API Reference Manual*
Stratisd Version 3.0

December 17, 2021

Contents

Asking Questions and Making Changes to this Document

This document can be foundin the stratis-docs repo, . 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 messages 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.

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 filesystems that 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

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

The `org.freedesktop.dbus.ObjectManager.InterfacesAdded` and `org.freedesktop.dbus.ObjectManager.InterfacesRemoved` signals are emitted as appropriate.

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.

Idempotency It is intended that all operations that can be requested via the D-Bus API be idempotent. By this is meant that if an operation is requested repeatedly, and there are no intervening actions, then the operation should succeed every time, or it should fail every time. Since the return values may encode information about actions taken these may differ between different invocations. However if an error is returned that should always be the same error. Lack of idempotency as defined here constitutes a bug.

Structure of the Return Value All methods return a return code and an accompanying message, with type signature "qs", and may also return data. If a method does return data, then the data is the first element in a triple, followed by the return code and message. The data may be a container type such as a struct or an array. Otherwise, if the method returns no data, the returned value is just a pair of the return code and message.

Duplicate Device Nodes in Method Arguments Some methods expect an array of device nodes as an argument. It is not considered an error if the array contains duplicate items; the items are reduced to a set. The order in which the devices are processed, for example, are added to an existing pool, is not guaranteed to be the same as the order in which they are passed as arguments.

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. This same scheme is used in the results of various D-Bus method and also for some properties.

UUID format UUIDs are in simple format, i.e., without hyphens.

Time format Times, such as the initialization time of a block device, are in RFC 3339 and ISO 8601 format.

Size format All sizes, whether as parameters, in results, or as properties, are specified in bytes and formatted as strings.

4 Stratis interface versioning

4.1 Versioning scheme

stratisd implements a versioned D-Bus interface.

4.1.1 Interface major versions

The versioning scheme is currently included in the bus name (for example, org.storage.stratis3) and all of the interface names (org.storage.stratis3.pool, org.storage.stratis3.filesystem, and so on). The interface version number corresponds to the major version number of stratisd.

4.1.2 Interface minor versions

The scheme allows supporting multiple interface revisions. For each minor version of stratisd a new revision of each existing interface is introduced, while prior interface revisions continue to be supported. The revision signifier is appended to the interface stem, e.g., org.storage.sratis3.pool.r0, org.storage.stratis3.pool.r1.

4.2 Interface breaking changes

If a user installs from source, there are some considerations when updating to a new major version. Because stratisd uses the system bus in D-Bus, D-Bus will only allow stratis to bind to the new interface name if the D-Bus policy file allowing this is updated. When a new major version is released this file must be edited or else stratis will fail to communicate using the CLI or the programmatic API over D-Bus.

5 Introspection Data

The introspection data for every kind of object stratisd supports, with accompanying comments describing the use of the parameters and the meaning of the return values is available via the following links:

Manager https://stratis-storage.github.io/manager.xml

blockdev https://stratis-storage.github.io/blockdev.xml

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

filesystem https://stratis-storage.github.io/filesystem.xml

pool https://stratis-storage.github.io/pool.xml