NAME

org.freedesktop.systemd1 − The D−Bus interface of systemd

INTRODUCTION

systemd(1) and its auxiliary daemons expose a number of APIs over D−Bus. This page only describes the various APIs exposed by the system and service manager itself. It does not cover the auxiliary daemons.

The service manager exposes a number of objects on the bus: one Manager object as a central entry point for clients along with individual objects for each unit and for each queued job. The unit objects implement a generic Unit interface as well as a type−specific interface. For example, service units implement both org.freedesktop.systemd1.Unit and org.freedesktop.system1.Service. The manager object can list unit and job objects or directly convert a unit name or job identifier to a bus path of the corresponding D−Bus object.

Properties exposing time values are usually encoded in microseconds (µs) on the bus, even if their corresponding settings in the unit files are in seconds.

PID 1 uses **polkit**[1] to allow access to privileged operations for unprivileged processes. Some operations (such as shutdown/reboot/suspend) are also available through the D−Bus API of logind, see **org.freedesktop.login1**(5).

THE MANAGER OBJECT

The main entry point object is available on the fixed **/org/freedesktop/systemd1** object path:

```
node /org/freedesktop/systemd1 {
  interface org.freedesktop.systemd1.Manager {
    methods:
      GetUnit(in  s name,
              out o unit);
      GetUnitByPID(in  u pid,
                   out o unit);
      GetUnitByInvocationID(in  ay invocation_id,
                            out o unit);
      GetUnitByControlGroup(in  s cgroup,
                            out o unit);
      LoadUnit(in  s name,
               out o unit);
      StartUnit(in  s name,
                in  s mode,
                out o job);
      StartUnitReplace(in  s old_unit,
                       in  s new_unit,
                       in  s mode,
                       out o job);
      StopUnit(in  s name,
               in  s mode,
               out o job);
      ReloadUnit(in  s name,
                 in  s mode,
                 out o job);
      RestartUnit(in  s name,
                  in  s mode,
                  out o job);
      TryRestartUnit(in  s name,
                     in  s mode,
                     out o job);
      ReloadOrRestartUnit(in  s name,
```

```
                    in  s mode,
                    out o job);
ReloadOrTryRestartUnit(in  s name,
                       in  s mode,
                       out o job);
EnqueueUnitJob(in  s name,
               in  s job_type,
               in  s job_mode,
               out u job_id,
               out o job_path,
               out s unit_id,
               out o unit_path,
               out s job_type,
               out a(uosos) affected_jobs);
KillUnit(in  s name,
         in  s whom,
         in  i signal);
CleanUnit(in  s name,
          in  as mask);
FreezeUnit(in  s name);
ThawUnit(in  s name);
ResetFailedUnit(in  s name);
SetUnitProperties(in  s name,
                  in  b runtime,
                  in  a(sv) properties);
BindMountUnit(in  s name,
              in  s source,
              in  s destination,
              in  b read_only,
              in  b mkdir);
MountImageUnit(in  s name,
               in  s source,
               in  s destination,
               in  b read_only,
               in  b mkdir,
               in  a(ss) options);
RefUnit(in  s name);
UnrefUnit(in  s name);
StartTransientUnit(in  s name,
                   in  s mode,
                   in  a(sv) properties,
                   in  a(sa(sv)) aux,
                   out o job);
GetUnitProcesses(in  s name,
                 out a(sus) processes);
AttachProcessesToUnit(in  s unit_name,
                      in  s subcgroup,
                      in  au pids);
AbandonScope(in  s name);
GetJob(in  u id,
       out o job);
GetJobAfter(in  u id,
            out a(usssoo) jobs);
GetJobBefore(in  u id,
```

```
                out a(usssoo) jobs);
        CancelJob(in  u id);
        ClearJobs();
        ResetFailed();
        SetShowStatus(in  s mode);
        ListUnits(out a(ssssssouso) units);
        ListUnitsFiltered(in  as states,
                          out a(ssssssouso) units);
        ListUnitsByPatterns(in  as states,
                            in  as patterns,
                            out a(ssssssouso) units);
        ListUnitsByNames(in  as names,
                         out a(ssssssouso) units);
        ListJobs(out a(usssoo) jobs);
        Subscribe();
        Unsubscribe();
        Dump(out s output);
        DumpByFileDescriptor(out h fd);
        Reload();
        Reexecute();
        Exit();
        Reboot();
        PowerOff();
        Halt();
        KExec();
        SwitchRoot(in  s new_root,
                   in  s init);
        SetEnvironment(in  as assignments);
        UnsetEnvironment(in  as names);
        UnsetAndSetEnvironment(in  as names,
                               in  as assignments);
        EnqueueMarkedJobs(out ao jobs);
        ListUnitFiles(out a(ss) unit_files);
        ListUnitFilesByPatterns(in  as states,
                                in  as patterns,
                                out a(ss) unit_files);
        GetUnitFileState(in  s file,
                         out s state);
        EnableUnitFiles(in  as files,
                        in  b runtime,
                        in  b force,
                        out b carries_install_info,
                        out a(sss) changes);
        DisableUnitFiles(in  as files,
                         in  b runtime,
                         out a(sss) changes);
        EnableUnitFilesWithFlags(in  as files,
                                 in  t flags,
                                 out b carries_install_info,
                                 out a(sss) changes);
        DisableUnitFilesWithFlags(in  as files,
                                  in  t flags,
                                  out a(sss) changes);
        ReenableUnitFiles(in  as files,
```

```
                    in  b runtime,
                    in  b force,
                    out b carries_install_info,
                    out a(sss) changes);
  LinkUnitFiles(in  as files,
                in  b runtime,
                in  b force,
                out a(sss) changes);
  PresetUnitFiles(in  as files,
                  in  b runtime,
                  in  b force,
                  out b carries_install_info,
                  out a(sss) changes);
  PresetUnitFilesWithMode(in  as files,
                          in  s mode,
                          in  b runtime,
                          in  b force,
                          out b carries_install_info,
                          out a(sss) changes);
  MaskUnitFiles(in  as files,
                in  b runtime,
                in  b force,
                out a(sss) changes);
  UnmaskUnitFiles(in  as files,
                  in  b runtime,
                  out a(sss) changes);
  RevertUnitFiles(in  as files,
                  out a(sss) changes);
  SetDefaultTarget(in  s name,
                   in  b force,
                   out a(sss) changes);
  GetDefaultTarget(out s name);
  PresetAllUnitFiles(in  s mode,
                     in  b runtime,
                     in  b force,
                     out a(sss) changes);
  AddDependencyUnitFiles(in  as files,
                         in  s target,
                         in  s type,
                         in  b runtime,
                         in  b force,
                         out a(sss) changes);
  GetUnitFileLinks(in  s name,
                   in  b runtime,
                   out as links);
  SetExitCode(in  y number);
  LookupDynamicUserByName(in  s name,
                          out u uid);
  LookupDynamicUserByUID(in  u uid,
                         out s name);
  GetDynamicUsers(out a(us) users);
signals:
  UnitNew(s id,
          o unit);
```

```
  UnitRemoved(s id,
              o unit);
  JobNew(u id,
         o job,
         s unit);
  JobRemoved(u id,
             o job,
             s unit,
             s result);
  StartupFinished(t firmware,
                  t loader,
                  t kernel,
                  t initrd,
                  t userspace,
                  t total);
  UnitFilesChanged();
  Reloading(b active);
properties:
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly s Version = '...';
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly s Features = '...';
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly s Virtualization = '...';
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly s Architecture = '...';
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly s Tainted = '...';
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly t FirmwareTimestamp = ...;
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly t FirmwareTimestampMonotonic = ...;
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly t LoaderTimestamp = ...;
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly t LoaderTimestampMonotonic = ...;
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly t KernelTimestamp = ...;
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly t KernelTimestampMonotonic = ...;
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly t InitRDTimestamp = ...;
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly t InitRDTimestampMonotonic = ...;
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly t UserspaceTimestamp = ...;
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly t UserspaceTimestampMonotonic = ...;
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly t FinishTimestamp = ...;
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly t FinishTimestampMonotonic = ...;
  @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
  readonly t SecurityStartTimestamp = ...;
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t SecurityStartTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t SecurityFinishTimestamp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t SecurityFinishTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t GeneratorsStartTimestamp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t GeneratorsStartTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t GeneratorsFinishTimestamp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t GeneratorsFinishTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t UnitsLoadStartTimestamp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t UnitsLoadStartTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t UnitsLoadFinishTimestamp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t UnitsLoadFinishTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t InitRDSecurityStartTimestamp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t InitRDSecurityStartTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t InitRDSecurityFinishTimestamp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t InitRDSecurityFinishTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t InitRDGeneratorsStartTimestamp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t InitRDGeneratorsStartTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t InitRDGeneratorsFinishTimestamp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t InitRDGeneratorsFinishTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t InitRDUnitsLoadStartTimestamp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t InitRDUnitsLoadStartTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t InitRDUnitsLoadFinishTimestamp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t InitRDUnitsLoadFinishTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
@org.freedesktop.systemd1.Privileged("true")
readwrite s LogLevel = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
@org.freedesktop.systemd1.Privileged("true")
readwrite s LogTarget = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly u NNames = ...;
```

```
readonly u NFailedUnits = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly u NJobs = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly u NInstalledJobs = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly u NFailedJobs = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly d Progress = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as Environment = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ConfirmSpawn = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b ShowStatus = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as UnitPath = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s DefaultStandardOutput = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s DefaultStandardError = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
@org.freedesktop.systemd1.Privileged("true")
readwrite t RuntimeWatchdogUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
@org.freedesktop.systemd1.Privileged("true")
readwrite t RebootWatchdogUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
@org.freedesktop.systemd1.Privileged("true")
readwrite t KExecWatchdogUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
@org.freedesktop.systemd1.Privileged("true")
readwrite b ServiceWatchdogs = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ControlGroup = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s SystemState = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly y ExitCode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultTimerAccuracyUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultTimeoutStartUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultTimeoutStopUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t DefaultTimeoutAbortUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultRestartUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultStartLimitIntervalUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u DefaultStartLimitBurst = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
readonly b DefaultCPUAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b DefaultBlockIOAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b DefaultMemoryAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b DefaultTasksAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitCPU = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitCPUSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitFSIZE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitFSIZESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitDATA = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitDATASoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitSTACK = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitSTACKSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitCORE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitCORESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitRSS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitRSSSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitNOFILE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitNOFILESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitAS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitASSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitNPROC = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitNPROCSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitMEMLOCK = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitMEMLOCKSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitLOCKS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitLOCKSSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t DefaultLimitSIGPENDING = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
        readonly t DefaultLimitSIGPENDINGSoft = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t DefaultLimitMSGQUEUE = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t DefaultLimitMSGQUEUESoft = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t DefaultLimitNICE = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t DefaultLimitNICESoft = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t DefaultLimitRTPRIO = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t DefaultLimitRTPRIOSoft = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t DefaultLimitRTTIME = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t DefaultLimitRTTIMESoft = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly t DefaultTasksMax = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t TimerSlackNSec = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s DefaultOOMPolicy = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s CtrlAltDelBurstAction = '...';
    };
    interface org.freedesktop.DBus.Peer { ... };
    interface org.freedesktop.DBus.Introspectable { ... };
    interface org.freedesktop.DBus.Properties { ... };
};
```

Methods

Note that many of the methods exist twice: once on the Manager object and once on the respective unit objects. This is to optimize access times so that methods that belong to unit objects do not have to be called with a resolved unit path, but can be called with only the unit id, too.

GetUnit() may be used to get the unit object path for a unit name. It takes the unit name and returns the object path. If a unit has not been loaded yet by this name this method will fail.

GetUnitByPID() may be used to get the unit object path of the unit a process ID belongs to. It takes a UNIX PID and returns the object path. The PID must refer to an existing system process.

LoadUnit() is similar to **GetUnit()** but will load the unit from disk if possible.

StartUnit() enqueues a start job and possibly depending jobs. It takes the unit to activate and a mode string as arguments. The mode needs to be one of "replace", "fail", "isolate", "ignore−dependencies", or "ignore−requirements". If "replace", the method will start the unit and its dependencies, possibly replacing already queued jobs that conflict with it. If "fail", the method will start the unit and its dependencies, but will fail if this would change an already queued job. If "isolate", the method will start the unit in question and terminate all units that aren't dependencies of it. If "ignore−dependencies", it will start a unit but ignore all its dependencies. If "ignore−requirements", it will start a unit but only ignore the requirement dependencies. It is not recommended to make use of the latter two options. On completion, this method returns the newly created job object.

StartUnitReplace() is similar to **StartUnit()** but replaces a job that is queued for one unit by a job for another unit.

StopUnit() is similar to **StartUnit()** but stops the specified unit rather than starting it. Note that the "isolate" mode is invalid for this method.

ReloadUnit(), **RestartUnit()**, **TryRestartUnit()**, **ReloadOrRestartUnit()**, or **ReloadOrTryRestartUnit()** may be used to restart and/or reload a unit. These methods take similar arguments as **StartUnit()**. Reloading is done only if the unit is already running and fails otherwise. If a service is restarted that isn't running, it will be started unless the "Try" flavor is used in which case a service that isn't running is not affected by the restart. The "ReloadOrRestart" flavors attempt a reload if the unit supports it and use a restart otherwise.

EnqueueMarkedJobs() creates reload/restart jobs for units which have been appropriately marked, see *Marks* property above. This is equivalent to calling **TryRestartUnit()** or **ReloadOrTryRestartUnit()** for the marked units.

BindMountUnit() can be used to bind mount new files or directories into a running service mount namespace.

MountImageUnit() can be used to mount new images into a running service mount namespace.

KillUnit() may be used to kill (i.e. send a signal to) all processes of a unit. It takes the unit *name*, an enum *who* and a UNIX *signal* number to send. The *who* enum is one of "main", "control" or "all". If "main", only the main process of the unit is killed. If "control", only the control process of the unit is killed. If "all", all processes are killed. A "control" process is for example a process that is configured via *ExecStop=* and is spawned in parallel to the main daemon process in order to shut it down.

GetJob() returns the job object path for a specific job, identified by its id.

CancelJob() cancels a specific job identified by its numeric ID. This operation is also available in the **Cancel()** method of Job objects (see below) and exists primarily to reduce the necessary round trips to execute this operation. Note that this will not have any effect on jobs whose execution has already begun.

ClearJobs() flushes the job queue, removing all jobs that are still queued. Note that this does not have any effect on jobs whose execution has already begun. It only flushes jobs that are queued and have not yet begun execution.

ResetFailedUnit() resets the "failed" state of a specific unit.

ResetFailed() resets the "failed" state of all units.

ListUnits() returns an array of all currently loaded units. Note that units may be known by multiple names at the same name, and hence there might be more unit names loaded than actual units behind them. The array consists of structures with the following elements:

- The primary unit name as string
- The human readable description string
- The load state (i.e. whether the unit file has been loaded successfully)
- The active state (i.e. whether the unit is currently started or not)
- The sub state (a more fine−grained version of the active state that is specific to the unit type, which the active state is not)
- A unit that is being followed in its state by this unit, if there is any, otherwise the empty string.
- The unit object path
- If there is a job queued for the job unit, the numeric job id, 0 otherwise
- The job type as string
- The job object path

ListJobs() returns an array with all currently queued jobs. Returns an array consisting of structures with the following elements:

- The numeric job id
- The primary unit name for this job
- The job type as string
- The job state as string
- The job object path
- The unit object path

Subscribe() enables most bus signals to be sent out. Clients which are interested in signals need to call this method. Signals are only sent out if at least one client invoked this method. **Unsubscribe()** reverts the signal subscription that **Subscribe()** implements. It is not necessary to invoke **Unsubscribe()** as clients are tracked. Signals are no longer sent out as soon as all clients which previously asked for **Subscribe()** either closed their connection to the bus or invoked **Unsubscribe()**.

Reload() may be invoked to reload all unit files.

Reexecute() may be invoked to reexecute the main manager process. It will serialize its state, reexecute, and deserizalize the state again. This is useful for upgrades and is a more comprehensive version of **Reload()**.

Exit() may be invoked to ask the manager to exit. This is not available for the system manager and is useful only for user session managers.

Reboot(), **PowerOff()**, **Halt()**, or **KExec()** may be used to ask for immediate reboot, powering down, halt or kexec based reboot of the system. Note that this does not shut down any services and immediately transitions into the reboot process. These functions are normally only called as the last step of shutdown and should not be called directly. To shut down the machine, it is generally a better idea to invoke **Reboot()** or **PowerOff()** on the systemd−logind manager object; see **org.freedesktop.login1**(5) for more information.

SwitchRoot() may be used to transition to a new root directory. This is intended to be used by initial RAM disks. The method takes two arguments: the new root directory (which needs to be specified) and an init binary path (which may be left empty, in which case it is automatically searched for). The state of the system manager will be serialized before the transition. After the transition, the manager binary on the main system is invoked and replaces the old PID 1. All state will then be deserialized.

SetEnvironment() may be used to alter the environment block that is passed to all spawned processes. It

takes a string array of environment variable assignments. Any previously set environment variables will be overridden.

UnsetEnvironment() may be used to unset environment variables. It takes a string array of environment variable names. All variables specified will be unset (if they have been set previously) and no longer be passed to all spawned processes. This method has no effect for variables that were previously not set, but will not fail in that case.

UnsetAndSetEnvironment() is a combination of **UnsetEnvironment()** and **SetEnvironment()**. It takes two lists. The first list contains variables to unset, the second one contains assignments to set. If a variable is listed in both, the variable is set after this method returns, i.e. the set list overrides the unset list.

ListUnitFiles() returns an array of unit names and their enablement status. Note that **ListUnit()** returns a list of units currently loaded into memory, while **ListUnitFiles()** returns a list of unit *files* that were found on disk. Note that while most units are read directly from a unit file with the same name, some units are not backed by files and some files (templates) cannot directly be loaded as units but need to be instantiated instead.

GetUnitFileState() returns the current enablement status of a specific unit file.

EnableUnitFiles() may be used to enable one or more units in the system (by creating symlinks to them in /etc/ or /run/). It takes a list of unit files to enable (either just file names or full absolute paths if the unit files are residing outside the usual unit search paths) and two booleans: the first controls whether the unit shall be enabled for runtime only (true, /run/), or persistently (false, /etc/). The second one controls whether symlinks pointing to other units shall be replaced if necessary. This method returns one boolean and an array of the changes made. The boolean signals whether the unit files contained any enablement information (i.e. an [Install]) section. The changes array consists of structures with three strings: the type of the change (one of "symlink" or "unlink"), the file name of the symlink and the destination of the symlink. Note that most of the following calls return a changes list in the same format.

Similarly, **DisableUnitFiles()** disables one or more units in the system, i.e. removes all symlinks to them in /etc/ and /run/.

The **EnableUnitFilesWithFlags()** and **DisableUnitFilesWithFlags()** take in options as flags instead of booleans to allow for extendability, defined as follows:

```
#define SD_SYSTEMD_UNIT_RUNTIME  (UINT64_C(1) << 0)
#define SD_SYSTEMD_UNIT_FORCE    (UINT64_C(1) << 1)
#define SD_SYSTEMD_UNIT_PORTABLE (UINT64_C(1) << 2)
```

SD_SYSTEMD_UNIT_RUNTIME will enable or disable the unit for runtime only, *SD_SYSTEMD_UNIT_FORCE* controls whether symlinks pointing to other units shall be replaced if necessary. *SD_SYSTEMD_UNIT_PORTABLE* will add or remove the symlinks in /etc/systemd/system.attached and /run/systemd/system.attached.

Similarly, **ReenableUnitFiles()** applies the changes to one or more units that would result from disabling and enabling the unit quickly one after the other in an atomic fashion. This is useful to apply updated [Install] information contained in unit files.

Similarly, **LinkUnitFiles()** links unit files (that are located outside of the usual unit search paths) into the unit search path.

Similarly, **PresetUnitFiles()** enables/disables one or more unit files according to the preset policy. See **systemd.preset**(7) for more information.

Similarly, **MaskUnitFiles()** masks unit files and **UnmaskUnitFiles()** unmasks them again.

SetDefaultTarget() changes the default.target link. See **bootup**(7) for more information.

GetDefaultTarget() retrieves the name of the unit to which default.target is aliased.

SetUnitProperties() may be used to modify certain unit properties at runtime. Not all properties may be

changed at runtime, but many resource management settings (primarily those listed in **systemd.resource-control**(5)) may. The changes are applied instantly and stored on disk for future boots, unless *runtime* is true, in which case the settings only apply until the next reboot. *name* is the name of the unit to modify. *properties* are the settings to set, encoded as an array of property name and value pairs. Note that this is not a dictionary! Also note that when setting array properties with this method usually results in appending to the pre−configured array. To reset the configured arrays, set the property to an empty array first and then append to it.

StartTransientUnit() may be used to create and start a transient unit which will be released as soon as it is not running or referenced anymore or the system is rebooted. *name* is the unit name including its suffix and must be unique. *mode* is the same as in **StartUnit()**, *properties* contains properties of the unit, specified like in **SetUnitProperties()**. *aux* is currently unused and should be passed as an empty array. See the **New Control Group Interface**[2] for more information how to make use of this functionality for resource control purposes.

Signals

Note that most signals are sent out only after **Subscribe()** has been invoked by at least one client. Make sure to invoke this method when subscribing to these signals!

UnitNew() and **UnitRemoved()** are sent out each time a new unit is loaded or unloaded. Note that this has little to do with whether a unit is available on disk or not, and simply reflects the units that are currently loaded into memory. The signals take two parameters: the primary unit name and the object path.

JobNew() and **JobRemoved()** are sent out each time a new job is queued or dequeued. Both signals take the numeric job ID, the bus path and the primary unit name for this job as arguments. **JobRemoved()** also includes a result string which is one of "done", "canceled", "timeout", "failed", "dependency", or "skipped". "done" indicates successful execution of a job. "canceled" indicates that a job has been canceled (via **CancelJob()** above) before it finished execution (this doesn't necessarily mean though that the job operation is actually cancelled too, see above). "timeout" indicates that the job timeout was reached. "failed" indicates that the job failed. "dependency" indicates that a job this job depended on failed and the job hence was removed as well. "skipped" indicates that a job was skipped because it didn't apply to the unit's current state.

StartupFinished() is sent out when startup finishes. It carries six microsecond timespan values, each indicating how much boot time has been spent in the firmware (if known), in the boot loader (if known), in the kernel initialization phase, in the initrd (if known), in userspace and in total. These values may also be calculated from the *FirmwareTimestampMonotonic*, *LoaderTimestampMonotonic*, *InitRDTimestampMonotonic*, *UserspaceTimestampMonotonic*, and *FinishTimestampMonotonic* properties (see below).

UnitFilesChanged() is sent out each time the list of enabled or masked unit files on disk have changed.

Reloading() is sent out immediately before a daemon reload is done (with the boolean parameter set to True) and after a daemon reload is completed (with the boolean parameter set to False). This may be used by UIs to optimize UI updates.

Properties

Most properties simply reflect the respective options in /etc/systemd/system.conf and the kernel command line.

The others:

Version encodes the version string of the running systemd instance. Note that the version string is purely informational. It should not be parsed and one may not assume the version to be formatted in any particular way. We take the liberty to change the versioning scheme at any time and it is not part of the public API.

Features encodes the features that have been enabled and disabled for this build. Enabled options are prefixed with +, disabled options with −.

Tainted encodes a couple of taint flags as a colon−separated list. When systemd detects it is running on a system with certain problems, it will set an appropriate taint flag. Taints may be used to lower the chance of bogus bug reports. The following taints are currently known: "split−usr", "mtab−not−symlink",

"cgroups−missing", "local−hwclock". "split−usr" is set if /usr/ is not pre−mounted when systemd is first invoked. See **Booting Without /usr is Broken**[3] for details why this is bad. "mtab−not−symlink" indicates that /etc/mtab is not a symlink to /proc/self/mounts as required. "cgroups−missing" indicates that control groups have not been enabled in the kernel. "local−hwclock" indicates that the local RTC is configured to be in local time rather than UTC.

FirmwareTimestamp, *FirmwareTimestampMonotonic*, *LoaderTimestamp*, *LoaderTimestampMonotonic*, *KernelTimestamp*, *KernelTimestampMonotonic*, *InitRDTimestamp*, *InitRDTimestampMonotonic*, *UserspaceTimestamp*, *UserspaceTimestampMonotonic*, *FinishTimestamp*, and *FinishTimestampMonotonic* encode **CLOCK_REALTIME** and **CLOCK_MONOTONIC** microsecond timestamps taken when the firmware first began execution, when the boot loader first began execution, when the kernel first began execution, when the initrd first began execution, when the main systemd instance began execution and finally, when all queued startup jobs finished execution. These values are useful for determining boot−time performance. Note that as monotonic time begins with the kernel startup, the *KernelTimestampMonotonic* timestamp will always be 0 and *FirmwareTimestampMonotonic* and *LoaderTimestampMonotonic* are to be read as negative values. Also, not all fields are always available, depending on the used firmware, boot loader or initrd implementation. In these cases the respective pairs of timestamps are both 0, indicating that no data is available.

Similarly, the *SecurityStartTimestamp*, *GeneratorsStartTimestamp* and *LoadUnitTimestamp* (as well as their monotonic and stop counterparts) expose performance data for uploading the security policies to the kernel (such as the SELinux, IMA, or SMACK policies), for running the generator tools and for loading the unit files.

NNames encodes how many unit names are currently known. This only includes names of units that are currently loaded and can be more than the amount of actually loaded units since units may have more than one name.

NJobs encodes how many jobs are currently queued.

NInstalledJobs encodes how many jobs have ever been queued in total.

NFailedJobs encodes how many jobs have ever failed in total.

Progress encodes boot progress as a floating point value between 0.0 and 1.0. This value begins at 0.0 at early−boot and ends at 1.0 when boot is finished and is based on the number of executed and queued jobs. After startup, this field is always 1.0 indicating a finished boot.

Environment encodes the environment block passed to all executed services. It may be altered with bus calls such as **SetEnvironment()** (see above).

UnitPath encodes the currently active unit file search path. It is an array of file system paths encoded as strings.

Virtualization contains a short ID string describing the virtualization technology the system runs in. On bare−metal hardware this is the empty string. Otherwise, it contains an identifier such as "kvm", "vmware" and so on. For a full list of IDs see **systemd-detect-virt**(1). Note that only the "innermost" virtualization technology is exported here. This detects both full−machine virtualizations (VMs) and shared−kernel virtualization (containers).

Architecture contains a short ID string describing the architecture the systemd instance is running on. This follows the same vocabulary as *ConditionArchitectures=*.

ControlGroup contains the root control group path of this system manager. Note that the root path is encoded as the empty string here (not as "/"!), so that it can be appended to /sys/fs/cgroup/systemd easily. This value will be set to the empty string for the host instance and some other string for container instances.

Security

Read access is generally granted to all clients. Additionally, for unprivileged clients, some operations are allowed through the polkit privilege system. Operations which modify unit state (**StartUnit()**, **StopUnit()**, **KillUnit()**, **RestartUnit()** and similar, **SetProperty()**) require org.freedesktop.systemd1.manage−units. Operations which modify unit file enablement state (**EnableUnitFiles()**, **DisableUnitFiles()**,

EnableUnitFilesWithFlags(), **DisableUnitFilesWithFlags()**, **ReenableUnitFiles()**, **LinkUnitFiles()**, **PresetUnitFiles**, **MaskUnitFiles**, and similar) require org.freedesktop.systemd1.manage−unit−files. Operations which modify the exported environment (**SetEnvironment()**, **UnsetEnvironment()**, **UnsetAndSetEnvironment()**) require org.freedesktop.systemd1.set−environment. **Reload()** and **Reexecute()** require org.freedesktop.systemd1.reload−daemon.

UNIT OBJECTS

```
node /org/freedesktop/systemd1/unit/avahi_2ddaemon_2eservice {
 interface org.freedesktop.systemd1.Unit {
  methods:
   Start(in  s mode,
         out o job);
   Stop(in  s mode,
        out o job);
   Reload(in  s mode,
          out o job);
   Restart(in  s mode,
           out o job);
   TryRestart(in  s mode,
              out o job);
   ReloadOrRestart(in  s mode,
                   out o job);
   ReloadOrTryRestart(in  s mode,
                      out o job);
   EnqueueJob(in  s job_type,
              in  s job_mode,
              out u job_id,
              out o job_path,
              out s unit_id,
              out o unit_path,
              out s job_type,
              out a(uosos) affected_jobs);
   Kill(in  s whom,
        in  i signal);
   ResetFailed();
   SetProperties(in  b runtime,
                 in  a(sv) properties);
   Ref();
   Unref();
   Clean(in  as mask);
   Freeze();
   Thaw();
  properties:
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly s Id = '...';
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly as Names = ['...', ...];
   @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
   readonly s Following = '...';
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly as Requires = ['...', ...];
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly as Requisite = ['...', ...];
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly as Wants = ['...', ...];
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as BindsTo = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as PartOf = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as RequiredBy = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as RequisiteOf = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as WantedBy = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as BoundBy = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ConsistsOf = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as Conflicts = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ConflictedBy = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as Before = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as After = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as OnFailure = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as OnFailureOf = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as OnSuccess = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as OnSuccessOf = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as Triggers = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as TriggeredBy = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as PropagatesReloadTo = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ReloadPropagatedFrom = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as PropagatesStopTo = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as StopPropagatedFrom = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as JoinsNamespaceOf = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as SliceOf = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as RequiresMountsFor = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as Documentation = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s Description = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s LoadState = '...';
```

```
readonly s ActiveState = '...';
readonly s FreezerState = '...';
readonly s SubState = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s FragmentPath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s SourcePath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as DropInPaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s UnitFileState = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s UnitFilePreset = '...';
readonly t StateChangeTimestamp = ...;
readonly t StateChangeTimestampMonotonic = ...;
readonly t InactiveExitTimestamp = ...;
readonly t InactiveExitTimestampMonotonic = ...;
readonly t ActiveEnterTimestamp = ...;
readonly t ActiveEnterTimestampMonotonic = ...;
readonly t ActiveExitTimestamp = ...;
readonly t ActiveExitTimestampMonotonic = ...;
readonly t InactiveEnterTimestamp = ...;
readonly t InactiveEnterTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b CanStart = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b CanStop = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b CanReload = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b CanIsolate = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as CanClean = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b CanFreeze = ...;
readonly (uo) Job = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b StopWhenUnneeded = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b RefuseManualStart = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b RefuseManualStop = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b AllowIsolate = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b DefaultDependencies = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s OnSuccessJobMode = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s OnFailureJobMode = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b IgnoreOnIsolate = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b NeedDaemonReload = ...;
```

```
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly as Markers = ['...', ...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t JobTimeoutUSec = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t JobRunningTimeoutUSec = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s JobTimeoutAction = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s JobTimeoutRebootArgument = '...';
        readonly b ConditionResult = ...;
        readonly b AssertResult = ...;
        readonly t ConditionTimestamp = ...;
        readonly t ConditionTimestampMonotonic = ...;
        readonly t AssertTimestamp = ...;
        readonly t AssertTimestampMonotonic = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
        readonly a(sbbsi) Conditions = [...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
        readonly a(sbbsi) Asserts = [...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly (ss) LoadError = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b Transient = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b Perpetual = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t StartLimitIntervalUSec = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly u StartLimitBurst = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s StartLimitAction = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s FailureAction = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i FailureActionExitStatus = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s SuccessAction = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i SuccessActionExitStatus = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s RebootArgument = '...';
        readonly ay InvocationID = [...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s CollectMode = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly as Refs = ['...', ...];
    };
    interface org.freedesktop.DBus.Peer { ... };
    interface org.freedesktop.DBus.Introspectable { ... };
    interface org.freedesktop.DBus.Properties { ... };
};
```

Methods

Start(), **Stop()**, **Reload()**, **Restart()**, **TryRestart()**, **ReloadOrRestart()**, **ReloadOrTryRestart()**, **Kill()**, **ResetFailed()**, and **SetProperties()** implement the same operation as the respective methods on the Manager object (see above). However, these methods operate on the unit object and hence do not take a unit name parameter. Invoking the methods directly on the Manager object has the advantage of not requiring a **GetUnit()** call to get the unit object for a specific unit name. Calling the methods on the Manager object is hence a round trip optimization.

Properties

Id contains the primary name of the unit.

Names contains all names of the unit, including the primary name that is also exposed in *Id*.

Following either contains the empty string or contains the name of another unit that this unit follows in state. This is used for some device units which reflect the unit state machine of another unit, and which other unit this is might possibly change.

Requires, *RequiresOverridable*, *Requisite*, *RequisiteOverridable*, *Wants*, *BindsTo*, *RequiredBy*, *RequiredByOverridable*, *WantedBy*, *BoundBy*, *Conflicts*, *ConflictedBy*, *Before*, *After*, *OnFailure*, *Triggers*, *TriggeredBy*, *PropagatesReloadTo*, and *RequiresMountsFor* contain arrays which encode the dependencies and their inverse dependencies (where this applies) as configured in the unit file or determined automatically.

Description contains the human readable description string for the unit.

SourcePath contains the path to a configuration file this unit is automatically generated from in case it is not a native unit (in which case it contains the empty string). For example, all mount units generated from /etc/fstab have this field set to /etc/fstab.

Documentation contains a string array with URLs of documentation for this unit.

LoadState contains a state value that reflects whether the configuration file of this unit has been loaded. The following states are currently defined: "loaded", "error", and "masked". "loaded" indicates that the configuration was successfully loaded. "error" indicates that the configuration failed to load. The *LoadError* field (see below) contains information about the cause of this failure. "masked" indicates that the unit is currently masked out (i.e. symlinked to /dev/null or empty). Note that the *LoadState* is fully orthogonal to the *ActiveState* (see below) as units without valid loaded configuration might be active (because configuration might have been reloaded at a time where a unit was already active).

ActiveState contains a state value that reflects whether the unit is currently active or not. The following states are currently defined: "active", "reloading", "inactive", "failed", "activating", and "deactivating". "active" indicates that unit is active (obviously...). "reloading" indicates that the unit is active and currently reloading its configuration. "inactive" indicates that it is inactive and the previous run was successful or no previous run has taken place yet. "failed" indicates that it is inactive and the previous run was not successful (more information about the reason for this is available on the unit type specific interfaces, for example for services in the *Result* property, see below). "activating" indicates that the unit has previously been inactive but is currently in the process of entering an active state. Conversely "deactivating" indicates that the unit is currently in the process of deactivation.

SubState encodes states of the same state machine that *ActiveState* covers, but knows more fine−grained states that are unit−type−specific. Where *ActiveState* only covers six high−level states, *SubState* covers possibly many more low−level unit−type−specific states that are mapped to the six high−level states. Note that multiple low−level states might map to the same high−level state, but not vice versa. Not all high−level states have low−level counterparts on all unit types. At this point the low−level states are not documented here, and are more likely to be extended later on than the common high−level states explained above.

FragmentPath contains the unit file path this unit was read from, if there is one (if not, it contains the empty string).

UnitFileState encodes the install state of the unit file of *FragmentPath*. It currently knows the following states: "enabled", "enabled−runtime", "linked", "linked−runtime", "masked", "masked−runtime", "static", "disabled", and "invalid". "enabled" indicates that a unit file is permanently enabled. "enable−runtime"

indicates the unit file is only temporarily enabled and will no longer be enabled after a reboot (that means, it is enabled via /run/ symlinks, rather than /etc/). "linked" indicates that a unit is linked into /etc/ permanently. "linked−runtime" indicates that a unit is linked into /run/ temporarily (until the next reboot). "masked" indicates that the unit file is masked permanently. "masked−runtime" indicates that it is masked in /run/ temporarily (until the next reboot). "static" indicates that the unit is statically enabled, i.e. always enabled and doesn't need to be enabled explicitly. "invalid" indicates that it could not be determined whether the unit file is enabled.

InactiveExitTimestamp, *InactiveExitTimestampMonotonic*, *ActiveEnterTimestamp*, *ActiveEnterTimestampMonotonic*, *ActiveExitTimestamp*, *ActiveExitTimestampMonotonic*, *InactiveEnterTimestamp*, and *InactiveEnterTimestampMonotonic* contain **CLOCK_REALTIME** and **CLOCK_MONOTONIC** 64−bit microsecond timestamps of the last time a unit left the inactive state, entered the active state, exited the active state, or entered an inactive state. These are the points in time where the unit transitioned "inactive"/"failed" → "activating", "activating" → "active", "active" → "deactivating", and finally "deactivating" → "inactive"/"failed". The fields are 0 in case such a transition has not yet been recorded on this boot.

CanStart, *CanStop*, and *CanReload* encode as booleans whether the unit supports the start, stop or reload operations. Even if a unit supports such an operation, the client might not necessary have the necessary privileges to execute them.

CanIsolate encodes as a boolean whether the unit may be started in isolation mode.

Job encodes the job ID and job object path of the job currently scheduled or executed for this unit, if there is any. If no job is scheduled or executed, the job id field will be 0.

StopWhenUnneeded, *RefuseManualStart*, *RefuseManualStop*, *AllowIsolate*, *DefaultDependencies*, *OnFailureIsolate*, *IgnoreOnIsolate*, *IgnoreOnSnapshot* map directly to the corresponding configuration booleans in the unit file.

DefaultControlGroup contains the main control group of this unit as a string. This refers to a group in systemd's own "name=systemd" hierarchy, which systemd uses to watch and manipulate the unit and all its processes.

NeedDaemonReload is a boolean that indicates whether the configuration file this unit is loaded from (i.e. *FragmentPath* or *SourcePath*) has changed since the configuration was read and hence whether a configuration reload is recommended.

Markers is an array of string flags that can be set using **SetUnitProperties()** to indicate that the service should be reloaded or restarted. Currently known values are "needs−restart" and "needs−reload". Package scripts may use the first to mark units for later restart when a new version of the package is installed. Configuration management scripts may use the second to mark units for a later reload when the configuration is adjusted. Those flags are not set by the manager, except to unset as appropriate when when the unit is stopped, restarted, or reloaded.

JobTimeoutUSec maps directly to the corresponding configuration setting in the unit file.

ConditionTimestamp and *ConditionTimestampMonotonic* contain the **CLOCK_REALTIME**/**CLOCK_MONOTONIC** microsecond timestamps of the last time the configured conditions of the unit have been checked or 0 if they have never been checked. Conditions are checked when a unit is requested to start.

ConditionResult contains the condition result of the last time the configured conditions of this unit were checked.

Conditions contains all configured conditions of the unit. For each condition, five fields are given: condition type (e.g. *ConditionPathExists*), whether the condition is a trigger condition, whether the condition is reversed, the right hand side of the condition (e.g. the path in case of *ConditionPathExists*), and the status. The status can be 0, in which case the condition hasn't been checked yet, a positive value, in which case the condition passed, or a negative value, in which case the condition failed. Currently only 0, +1, and −1 are used, but additional values may be used in the future, retaining the meaning of zero/positive/negative values.

LoadError contains a pair of strings. If the unit failed to load (as encoded in *LoadState*, see above), then this will include a D−Bus error pair consisting of the error ID and an explanatory human readable string of what happened. If it loaded successfully, this will be a pair of empty strings.

Transient contains a boolean that indicates whether the unit was created as a transient unit (i.e. via **CreateTransientUnit()** on the manager object).

Security

Similarly to methods on the Manager object, read−only access is allowed for everyone. All operations are allowed for clients with the **CAP_SYS_ADMIN** capability or when the org.freedesktop.systemd1.manage−units privilege is granted by polkit.

SERVICE UNIT OBJECTS

All service unit objects implement the org.freedesktop.systemd1.Service interface (described here) in addition to the generic org.freedesktop.systemd1.Unit interface (see above).

```
node /org/freedesktop/systemd1/unit/avahi_2ddaemon_2eservice {
 interface org.freedesktop.systemd1.Service {
  methods:
   BindMount(in  s source,
             in  s destination,
             in  b read_only,
             in  b mkdir);
   MountImage(in  s source,
              in  s destination,
              in  b read_only,
              in  b mkdir,
              in  a(ss) options);
   GetProcesses(out a(sus) processes);
   AttachProcesses(in  s subcgroup,
                   in  au pids);
  properties:
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly s Type = '...';
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly s Restart = '...';
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly s PIDFile = '...';
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly s NotifyAccess = '...';
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly t RestartUSec = ...;
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly t TimeoutStartUSec = ...;
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly t TimeoutStopUSec = ...;
   @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
   readonly t TimeoutAbortUSec = ...;
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly s TimeoutStartFailureMode = '...';
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly s TimeoutStopFailureMode = '...';
   @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
   readonly t RuntimeMaxUSec = ...;
   @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
   readonly t WatchdogUSec = ...;
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t WatchdogTimestamp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t WatchdogTimestampMonotonic = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b RootDirectoryStartOnly = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b RemainAfterExit = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b GuessMainPID = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (aiai) RestartPreventExitStatus = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (aiai) RestartForceExitStatus = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (aiai) SuccessExitStatus = ...;
readonly u MainPID = ...;
readonly u ControlPID = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s BusName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u FileDescriptorStoreMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly u NFileDescriptorStore = ...;
readonly s StatusText = '...';
readonly i StatusErrno = ...;
readonly s Result = '...';
readonly s ReloadResult = '...';
readonly s CleanResult = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s USBFunctionDescriptors = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s USBFunctionStrings = '...';
readonly u UID = ...;
readonly u GID = ...;
readonly u NRestarts = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s OOMPolicy = '...';
readonly t ExecMainStartTimestamp = ...;
readonly t ExecMainStartTimestampMonotonic = ...;
readonly t ExecMainExitTimestamp = ...;
readonly t ExecMainExitTimestampMonotonic = ...;
readonly u ExecMainPID = ...;
readonly i ExecMainCode = ...;
readonly i ExecMainStatus = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasbttttuii) ExecCondition = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasastttttuii) ExecConditionEx = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasbttttuii) ExecStartPre = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasastttttuii) ExecStartPreEx = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
```

```
readonly a(sasbttttuii) ExecStart = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasasttttuii) ExecStartEx = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasbttttuii) ExecStartPost = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasasttttuii) ExecStartPostEx = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasbttttuii) ExecReload = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasasttttuii) ExecReloadEx = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasbttttuii) ExecStop = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasasttttuii) ExecStopEx = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasbttttuii) ExecStopPost = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasasttttuii) ExecStopPostEx = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s Slice = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ControlGroup = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryCurrent = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryAvailable = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUUsageNSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay EffectiveCPUs = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay EffectiveMemoryNodes = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t TasksCurrent = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPIngressBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPIngressPackets = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPEgressBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPEgressPackets = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOReadBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOReadOperations = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOWriteBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOWriteOperations = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b Delegate = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
```

```
readonly as DelegateControllers = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b CPUAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupCPUWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUShares = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupCPUShares = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUQuotaPerSecUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUQuotaPeriodUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay AllowedCPUs = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay AllowedMemoryNodes = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b IOAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IODeviceWeight = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOReadBandwidthMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOWriteBandwidthMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOReadIOPSMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOWriteIOPSMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IODeviceLatencyTargetUSec = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b BlockIOAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t BlockIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupBlockIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIODeviceWeight = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIOReadBandwidth = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIOWriteBandwidth = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b MemoryAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t DefaultMemoryLow = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
```

```
readonly t DefaultMemoryMin = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryMin = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryLow = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryHigh = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemorySwapMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryLimit = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s DevicePolicy = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(ss) DeviceAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b TasksAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t TasksMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b IPAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iayu) IPAddressAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iayu) IPAddressDeny = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as IPIngressFilterPath = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as IPEgressFilterPath = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as DisableControllers = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMSwap = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMMemoryPressure = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly u ManagedOOMMemoryPressureLimit = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMPreference = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(ss) BPFProgram = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iiqq) SocketBindAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iiqq) SocketBindDeny = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as Environment = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(sb) EnvironmentFiles = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as PassEnvironment = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
readonly as UnsetEnvironment = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u UMask = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCPU = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCPUSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitFSIZE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitFSIZESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitDATA = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitDATASoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSTACK = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSTACKSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCORE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCORESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRSS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRSSSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNOFILE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNOFILESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitAS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitASSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNPROC = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNPROCSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMEMLOCK = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMEMLOCKSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitLOCKS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitLOCKSSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSIGPENDING = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSIGPENDINGSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMSGQUEUE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
readonly t LimitMSGQUEUESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNICE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNICESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTPRIO = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTPRIOSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTTIME = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTTIMESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s WorkingDirectory = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootDirectory = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootImage = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ss) RootImageOptions = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay RootHash = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootHashPath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay RootHashSignature = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootHashSignaturePath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootVerity = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(sba(ss)) ExtensionImages = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ssba(ss)) MountImages = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i OOMScoreAdjust = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t CoredumpFilter = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i Nice = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i IOSchedulingClass = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i IOSchedulingPriority = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i CPUSchedulingPolicy = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i CPUSchedulingPriority = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay CPUAffinity = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b CPUAffinityFromNUMA = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
readonly i NUMAPolicy = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay NUMAMask = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t TimerSlackNSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b CPUSchedulingResetOnFork = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b NonBlocking = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardInput = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardInputFileDescriptorName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay StandardInputData = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardOutput = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardOutputFileDescriptorName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardError = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardErrorFileDescriptorName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s TTYPath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b TTYReset = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b TTYVHangup = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b TTYVTDisallocate = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SyslogPriority = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s SyslogIdentifier = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b SyslogLevelPrefix = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SyslogLevel = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SyslogFacility = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i LogLevelMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LogRateLimitIntervalUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u LogRateLimitBurst = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly aay LogExtraFields = [[...], ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s LogNamespace = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SecureBits = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
readonly t CapabilityBoundingSet = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t AmbientCapabilities = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s User = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s Group = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b DynamicUser = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b RemoveIPC = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(say) SetCredential = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ss) LoadCredential = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as SupplementaryGroups = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s PAMName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ReadWritePaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ReadOnlyPaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as InaccessiblePaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ExecPaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as NoExecPaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t MountFlags = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateTmp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateDevices = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectClock = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectKernelTunables = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectKernelModules = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectKernelLogs = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectControlGroups = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateNetwork = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateUsers = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateMounts = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateIPC = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
readonly s ProtectHome = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s ProtectSystem = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b SameProcessGroup = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s UtmpIdentifier = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s UtmpMode = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bs) SELinuxContext = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bs) AppArmorProfile = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bs) SmackProcessLabel = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b IgnoreSIGPIPE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b NoNewPrivileges = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bas) SystemCallFilter = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as SystemCallArchitectures = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SystemCallErrorNumber = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bas) SystemCallLog = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s Personality = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b LockPersonality = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bas) RestrictAddressFamilies = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RuntimeDirectoryPreserve = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u RuntimeDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as RuntimeDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u StateDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as StateDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u CacheDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as CacheDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u LogsDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as LogsDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u ConfigurationDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
    readonly as ConfigurationDirectory = ['...', ...];
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly t TimeoutCleanUSec = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly b MemoryDenyWriteExecute = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly b RestrictRealtime = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly b RestrictSUIDSGID = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly t RestrictNamespaces = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly a(ssbt) BindPaths = [...];
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly a(ssbt) BindReadOnlyPaths = [...];
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly a(ss) TemporaryFileSystem = [...];
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly b MountAPIVFS = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly s KeyringMode = '...';
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly s ProtectProc = '...';
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly s ProcSubset = '...';
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly b ProtectHostname = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly s NetworkNamespacePath = '...';
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly s IPCNamespacePath = '...';
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly s KillMode = '...';
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly i KillSignal = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly i RestartKillSignal = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly i FinalKillSignal = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly b SendSIGKILL = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly b SendSIGHUP = ...;
    @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
    readonly i WatchdogSignal = ...;
  };
  interface org.freedesktop.DBus.Peer { ... };
  interface org.freedesktop.DBus.Introspectable { ... };
  interface org.freedesktop.DBus.Properties { ... };
  interface org.freedesktop.systemd1.Unit { ... };
};
```

Methods

BindMount() and **MountImage()** implement the same operations as the respective methods on the Manager object (see above). However, these methods operate on the service object and hence do not take a unit name parameter. Invoking the methods directly on the Manager object has the advantage of not requiring a **GetUnit()** call to get the unit object for a specific unit name. Calling the methods on the Manager object is hence a round trip optimization.

Properties

Most properties of the Service interface map directly to the corresponding settings in service unit files. For the sake of brevity, here's a list of all exceptions only:

TimeoutStartUSec, *TimeoutStopUSec* and *TimeoutAbortUSec* contain the start, stop and abort timeouts, in microseconds. Note the slight difference in naming when compared to the matching unit file settings (see **systemd.service**(7)): these bus properties strictly use microseconds (and thus are suffixed *...USec*) while the unit file settings default to a time unit of seconds (and thus are suffixed *...Sec*), unless a different unit is explicitly specified. This reflects that fact that internally the service manager deals in microsecond units only, and the bus properties are a relatively low−level (binary) concept exposing this. The unit file settings on the other hand are relatively high−level (string−based) concepts and thus support more user friendly time specifications which default to second time units but allow other units too, if specified.

WatchdogTimestamp and *WatchdogTimestampMonotonic* contain **CLOCK_REALTIME**/**CLOCK_MONOTONIC** microsecond timestamps of the last watchdog ping received from the service, or 0 if none was ever received.

ExecStartPre, *ExecStart*, *ExecStartPost*, *ExecReload*, *ExecStop*, and *ExecStop* are arrays of structures where each struct contains: the binary path to execute; an array with all arguments to pass to the executed command, starting with argument 0; a boolean whether it should be considered a failure if the process exits uncleanly; two pairs of **CLOCK_REALTIME**/**CLOCK_MONOTONIC** microsecond timestamps when the process began and finished running the last time, or 0 if it never ran or never finished running; the PID of the process, or 0 if it has not run yet; the exit code and status of the last run. This field hence maps more or less to the corresponding setting in the service unit file but is augmented with runtime data.

LimitCPU (and related properties) map more or less directly to the corresponding settings in the service unit files except that if they aren't set, their value is 18446744073709551615 (i.e. −1).

Capabilities contains the configured capabilities, as formatted with **cap_to_text**(3).

SecureBits, *CapabilityBoundingSet*, *MountFlags* also correspond to the configured settings of the unit files, but instead of being formatted as strings, they are encoded as the actual binary flags they are.

ExecMainStartTimestamp, *ExecMainStartTimestampMonotonic*, *ExecMainExitTimestamp*, *ExecMainExitTimestampMonotonic*, *ExecMainPID*, *ExecMainCode*, *ExecMainStatus* contain information about the main process of the service as far as it is known. This is often the same runtime information that is stored in *ExecStart*. However, it deviates for *Type=forking* services where the main process of the service is not forked off systemd directly. These fields either contain information of the last run of the process or of the current running process.

MainPID and *ControlPID* contain the main and control PID of the service. The main PID is the current main PID of the service and is 0 when the service currently has no main PID. The control PID is the PID of the current start/stop/reload process running and is 0 if no such process is currently running. That means that *ExecMainPID* and *MainPID* differ in the way that the latter immediately reflects whether a main process is currently running while the latter possible contains information collected from the last run even if the process is no longer around.

StatusText contains the status text passed to the service manager via a call to **sd_notify**(3). This may be used by services to inform the service manager about its internal state with a nice explanatory string.

Result encodes the execution result of the last run of the service. It is useful to determine the reason a service failed if it is in the "failed" state (see *ActiveState* above). The following values are currently known: "success" is set if the unit didn't fail. "resources" indicates that not enough resources were available to fork off and execute the service processes. "timeout" indicates that a timeout occurred while executing a service

operation. "exit−code" indicates that a service process exited with an unclean exit code. "signal" indicates that a service process exited with an uncaught signal. "core−dump" indicates that a service process exited uncleanly and dumped core. "watchdog" indicates that a service did not send out watchdog ping messages often enough. "start−limit" indicates that a service has been started too frequently in a specific time frame (as configured in *StartLimitInterval*, *StartLimitBurst*).

ControlGroup indicates the control group path the processes of this service unit are placed in.

The following properties map 1:1 to corresponding settings in the unit file: *RootDirectory RootImage RootImageOptions RootVerity RootHash RootHashSignature MountImages ExtensionImages* see systemd.exec(5) for their meaning.

MemoryAvailable indicates how much unused memory is available to the unit before the "MemoryMax" or "MemoryHigh" (whichever is lower) limit set by the cgroup memory controller is reached. It will take into consideration limits on all parent slices, other than the limits set on the unit itself.

SOCKET UNIT OBJECTS

```
node /org/freedesktop/systemd1/unit/avahi_2ddaemon_2esocket {
  interface org.freedesktop.systemd1.Socket {
    methods:
      GetProcesses(out a(sus) processes);
      AttachProcesses(in  s subcgroup,
                      in  au pids);
    properties:
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly s BindIPv6Only = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly u Backlog = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly t TimeoutUSec = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly s BindToDevice = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly s SocketUser = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly s SocketGroup = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly u SocketMode = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly u DirectoryMode = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly b Accept = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly b FlushPending = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly b Writable = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly b KeepAlive = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly t KeepAliveTimeUSec = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly t KeepAliveIntervalUSec = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly u KeepAliveProbes = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly t DeferAcceptUSec = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
readonly b NoDelay = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i Priority = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t ReceiveBuffer = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t SendBuffer = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i IPTOS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i IPTTL = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t PipeSize = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b FreeBind = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b Transparent = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b Broadcast = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PassCredentials = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PassSecurity = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PassPacketInfo = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s Timestamping = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b RemoveOnStop = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ss) Listen = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as Symlinks = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i Mark = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u MaxConnections = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u MaxConnectionsPerSource = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly x MessageQueueMaxMessages = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly x MessageQueueMessageSize = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s TCPCongestion = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ReusePort = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s SmackLabel = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s SmackLabelIPIn = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s SmackLabelIPOut = '...';
readonly u ControlPID = ...;
```

```
readonly s Result = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly u NConnections = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly u NAccepted = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly u NRefused = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s FileDescriptorName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SocketProtocol = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t TriggerLimitIntervalUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u TriggerLimitBurst = ...;
readonly u UID = ...;
readonly u GID = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasbttttuii) ExecStartPre = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasbttttuii) ExecStartPost = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasbttttuii) ExecStopPre = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
readonly a(sasbttttuii) ExecStopPost = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s Slice = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ControlGroup = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryCurrent = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryAvailable = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUUsageNSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay EffectiveCPUs = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay EffectiveMemoryNodes = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t TasksCurrent = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPIngressBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPIngressPackets = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPEgressBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPEgressPackets = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOReadBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOReadOperations = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
```

```
readonly t IOWriteBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOWriteOperations = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b Delegate = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as DelegateControllers = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b CPUAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupCPUWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUShares = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupCPUShares = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUQuotaPerSecUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUQuotaPeriodUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay AllowedCPUs = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay AllowedMemoryNodes = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b IOAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IODeviceWeight = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOReadBandwidthMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOWriteBandwidthMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOReadIOPSMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOWriteIOPSMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IODeviceLatencyTargetUSec = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b BlockIOAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t BlockIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupBlockIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIODeviceWeight = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIOReadBandwidth = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
```

```
readonly a(st) BlockIOWriteBandwidth = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b MemoryAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t DefaultMemoryLow = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t DefaultMemoryMin = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryMin = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryLow = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryHigh = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemorySwapMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryLimit = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s DevicePolicy = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(ss) DeviceAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b TasksAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t TasksMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b IPAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iayu) IPAddressAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iayu) IPAddressDeny = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as IPIngressFilterPath = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as IPEgressFilterPath = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as DisableControllers = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMSwap = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMMemoryPressure = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly u ManagedOOMMemoryPressureLimit = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMPreference = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(ss) BPFProgram = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iiqq) SocketBindAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iiqq) SocketBindDeny = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
readonly as Environment = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(sb) EnvironmentFiles = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as PassEnvironment = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as UnsetEnvironment = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u UMask = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCPU = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCPUSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitFSIZE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitFSIZESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitDATA = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitDATASoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSTACK = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSTACKSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCORE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCORESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRSS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRSSSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNOFILE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNOFILESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitAS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitASSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNPROC = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNPROCSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMEMLOCK = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMEMLOCKSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitLOCKS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitLOCKSSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
readonly t LimitSIGPENDING = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSIGPENDINGSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMSGQUEUE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMSGQUEUESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNICE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNICESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTPRIO = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTPRIOSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTTIME = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTTIMESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s WorkingDirectory = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootDirectory = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootImage = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ss) RootImageOptions = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay RootHash = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootHashPath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay RootHashSignature = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootHashSignaturePath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootVerity = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(sba(ss)) ExtensionImages = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ssba(ss)) MountImages = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i OOMScoreAdjust = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t CoredumpFilter = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i Nice = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i IOSchedulingClass = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i IOSchedulingPriority = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i CPUSchedulingPolicy = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
readonly i CPUSchedulingPriority = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay CPUAffinity = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b CPUAffinityFromNUMA = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i NUMAPolicy = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay NUMAMask = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t TimerSlackNSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b CPUSchedulingResetOnFork = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b NonBlocking = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardInput = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardInputFileDescriptorName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay StandardInputData = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardOutput = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardOutputFileDescriptorName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardError = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardErrorFileDescriptorName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s TTYPath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b TTYReset = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b TTYVHangup = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b TTYVTDisallocate = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SyslogPriority = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s SyslogIdentifier = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b SyslogLevelPrefix = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SyslogLevel = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SyslogFacility = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i LogLevelMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LogRateLimitIntervalUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u LogRateLimitBurst = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
readonly aay LogExtraFields = [[...], ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s LogNamespace = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SecureBits = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t CapabilityBoundingSet = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t AmbientCapabilities = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s User = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s Group = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b DynamicUser = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b RemoveIPC = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(say) SetCredential = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ss) LoadCredential = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as SupplementaryGroups = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s PAMName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ReadWritePaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ReadOnlyPaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as InaccessiblePaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ExecPaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as NoExecPaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t MountFlags = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateTmp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateDevices = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectClock = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectKernelTunables = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectKernelModules = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectKernelLogs = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectControlGroups = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateNetwork = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
readonly b PrivateUsers = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateMounts = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateIPC = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s ProtectHome = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s ProtectSystem = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b SameProcessGroup = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s UtmpIdentifier = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s UtmpMode = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bs) SELinuxContext = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bs) AppArmorProfile = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bs) SmackProcessLabel = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b IgnoreSIGPIPE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b NoNewPrivileges = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bas) SystemCallFilter = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as SystemCallArchitectures = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SystemCallErrorNumber = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bas) SystemCallLog = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s Personality = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b LockPersonality = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bas) RestrictAddressFamilies = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RuntimeDirectoryPreserve = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u RuntimeDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as RuntimeDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u StateDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as StateDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u CacheDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as CacheDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
```

```
        readonly u LogsDirectoryMode = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly as LogsDirectory = ['...', ...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly u ConfigurationDirectoryMode = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly as ConfigurationDirectory = ['...', ...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t TimeoutCleanUSec = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b MemoryDenyWriteExecute = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b RestrictRealtime = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b RestrictSUIDSGID = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t RestrictNamespaces = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly a(ssbt) BindPaths = [...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly a(ssbt) BindReadOnlyPaths = [...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly a(ss) TemporaryFileSystem = [...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b MountAPIVFS = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s KeyringMode = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s ProtectProc = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s ProcSubset = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b ProtectHostname = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s NetworkNamespacePath = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s IPCNamespacePath = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly s KillMode = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i KillSignal = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i RestartKillSignal = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i FinalKillSignal = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b SendSIGKILL = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b SendSIGHUP = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i WatchdogSignal = ...;
    };
    interface org.freedesktop.DBus.Peer { ... };
    interface org.freedesktop.DBus.Introspectable { ... };
```

```
  interface org.freedesktop.DBus.Properties { ... };
  interface org.freedesktop.systemd1.Unit { ... };
};
```

Properties

Most of the properties map directly to the corresponding settings in socket unit files. As socket units can include *ExecStartPre* (and similar) fields which contain information about processes to execute. They also share most of the fields related to the execution context that Service objects expose (see above).

In addition to these properties there are the following:

NAccepted contains the accumulated number of connections ever accepted on this socket. This only applies to sockets with *Accept* set to "yes", i.e. those where systemd is responsible for accepted connections.

Similarly *NConnections* contains the number of currently open connections on this socket. It only applies only to socket units with *Accept* set to "yes".

Result encodes the reason why a socket unit failed if it is in the "failed" state (see *ActiveState* above). The values "success", "resources", "timeout", "exit−code", "signal" and "core−dump" have the same meaning as they have for the corresponding field of service units (see above). In addition to that, the value "service−failed−permanent" indicates that the service of this socket failed continuously.

FlushPending specifies whether to flush the socket just before entering the listening state. This setting only applies to sockets with *Accept*= set to "no".

TARGET UNIT OBJECTS

```
node /org/freedesktop/systemd1/unit/basic_2etarget {
  interface org.freedesktop.systemd1.Target {
  };
  interface org.freedesktop.DBus.Peer { ... };
  interface org.freedesktop.DBus.Introspectable { ... };
  interface org.freedesktop.DBus.Properties { ... };
  interface org.freedesktop.systemd1.Unit { ... };
};
```

Target units have neither type−specific methods nor properties.

DEVICE UNIT OBJECTS

All device unit objects implement the org.freedesktop.systemd1.Device interface (described here) in addition to the generic org.freedesktop.systemd1.Unit interface (see above).

```
node /org/freedesktop/systemd1/unit/dev_2dttyS0_2edevice {
  interface org.freedesktop.systemd1.Device {
    properties:
      readonly s SysFSPath = '...';
  };
  interface org.freedesktop.DBus.Peer { ... };
  interface org.freedesktop.DBus.Introspectable { ... };
  interface org.freedesktop.DBus.Properties { ... };
  interface org.freedesktop.systemd1.Unit { ... };
};
```

Properties

Device units only expose a single type−specific property:

SysFSPath contains the sysfs path of the kernel device this object corresponds to.

MOUNT UNIT OBJECTS

All mount unit objects implement the org.freedesktop.systemd1.Mount interface (described here) in addition to the generic org.freedesktop.systemd1.Unit interface (see above).

```
node /org/freedesktop/systemd1/unit/home_2emount {
  interface org.freedesktop.systemd1.Mount {
    methods:
      GetProcesses(out a(sus) processes);
      AttachProcesses(in  s subcgroup,
                      in  au pids);
    properties:
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly s Where = '...';
      readonly s What = '...';
      readonly s Options = '...';
      readonly s Type = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly t TimeoutUSec = ...;
      readonly u ControlPID = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly u DirectoryMode = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly b SloppyOptions = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly b LazyUnmount = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly b ForceUnmount = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly b ReadWriteOnly = ...;
      readonly s Result = '...';
      readonly u UID = ...;
      readonly u GID = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
      readonly a(sasbttttuii) ExecMount = [...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
      readonly a(sasbttttuii) ExecUnmount = [...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
      readonly a(sasbttttuii) ExecRemount = [...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly s Slice = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly s ControlGroup = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t MemoryCurrent = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t MemoryAvailable = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t CPUUsageNSec = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly ay EffectiveCPUs = [...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly ay EffectiveMemoryNodes = [...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t TasksCurrent = ...;
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPIngressBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPIngressPackets = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPEgressBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPEgressPackets = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOReadBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOReadOperations = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOWriteBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOWriteOperations = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b Delegate = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as DelegateControllers = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b CPUAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupCPUWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUShares = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupCPUShares = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUQuotaPerSecUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUQuotaPeriodUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay AllowedCPUs = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay AllowedMemoryNodes = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b IOAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IODeviceWeight = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOReadBandwidthMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOWriteBandwidthMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOReadIOPSMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOWriteIOPSMax = [...];
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IODeviceLatencyTargetUSec = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b BlockIOAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t BlockIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupBlockIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIODeviceWeight = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIOReadBandwidth = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIOWriteBandwidth = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b MemoryAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t DefaultMemoryLow = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t DefaultMemoryMin = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryMin = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryLow = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryHigh = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemorySwapMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryLimit = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s DevicePolicy = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(ss) DeviceAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b TasksAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t TasksMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b IPAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iayu) IPAddressAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iayu) IPAddressDeny = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as IPIngressFilterPath = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as IPEgressFilterPath = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as DisableControllers = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMSwap = '...';
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMMemoryPressure = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly u ManagedOOMMemoryPressureLimit = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMPreference = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(ss) BPFProgram = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iiqq) SocketBindAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iiqq) SocketBindDeny = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as Environment = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(sb) EnvironmentFiles = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as PassEnvironment = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as UnsetEnvironment = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u UMask = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCPU = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCPUSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitFSIZE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitFSIZESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitDATA = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitDATASoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSTACK = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSTACKSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCORE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCORESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRSS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRSSSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNOFILE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNOFILESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitAS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitASSoft = ...;
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNPROC = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNPROCSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMEMLOCK = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMEMLOCKSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitLOCKS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitLOCKSSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSIGPENDING = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSIGPENDINGSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMSGQUEUE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMSGQUEUESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNICE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNICESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTPRIO = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTPRIOSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTTIME = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTTIMESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s WorkingDirectory = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootDirectory = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootImage = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ss) RootImageOptions = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay RootHash = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootHashPath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay RootHashSignature = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootHashSignaturePath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootVerity = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(sba(ss)) ExtensionImages = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ssba(ss)) MountImages = [...];
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i OOMScoreAdjust = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t CoredumpFilter = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i Nice = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i IOSchedulingClass = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i IOSchedulingPriority = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i CPUSchedulingPolicy = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i CPUSchedulingPriority = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay CPUAffinity = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b CPUAffinityFromNUMA = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i NUMAPolicy = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay NUMAMask = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t TimerSlackNSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b CPUSchedulingResetOnFork = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b NonBlocking = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardInput = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardInputFileDescriptorName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay StandardInputData = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardOutput = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardOutputFileDescriptorName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardError = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardErrorFileDescriptorName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s TTYPath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b TTYReset = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b TTYVHangup = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b TTYVTDisallocate = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SyslogPriority = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s SyslogIdentifier = '...';
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b SyslogLevelPrefix = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SyslogLevel = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SyslogFacility = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i LogLevelMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LogRateLimitIntervalUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u LogRateLimitBurst = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly aay LogExtraFields = [[...], ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s LogNamespace = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SecureBits = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t CapabilityBoundingSet = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t AmbientCapabilities = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s User = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s Group = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b DynamicUser = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b RemoveIPC = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(say) SetCredential = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ss) LoadCredential = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as SupplementaryGroups = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s PAMName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ReadWritePaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ReadOnlyPaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as InaccessiblePaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ExecPaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as NoExecPaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t MountFlags = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateTmp = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateDevices = ...;
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectClock = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectKernelTunables = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectKernelModules = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectKernelLogs = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectControlGroups = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateNetwork = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateUsers = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateMounts = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateIPC = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s ProtectHome = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s ProtectSystem = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b SameProcessGroup = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s UtmpIdentifier = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s UtmpMode = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bs) SELinuxContext = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bs) AppArmorProfile = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bs) SmackProcessLabel = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b IgnoreSIGPIPE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b NoNewPrivileges = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bas) SystemCallFilter = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as SystemCallArchitectures = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SystemCallErrorNumber = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bas) SystemCallLog = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s Personality = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b LockPersonality = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bas) RestrictAddressFamilies = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RuntimeDirectoryPreserve = '...';
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u RuntimeDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as RuntimeDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u StateDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as StateDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u CacheDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as CacheDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u LogsDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as LogsDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u ConfigurationDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ConfigurationDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t TimeoutCleanUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b MemoryDenyWriteExecute = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b RestrictRealtime = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b RestrictSUIDSGID = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t RestrictNamespaces = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ssbt) BindPaths = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ssbt) BindReadOnlyPaths = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ss) TemporaryFileSystem = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b MountAPIVFS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s KeyringMode = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s ProtectProc = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s ProcSubset = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectHostname = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s NetworkNamespacePath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s IPCNamespacePath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s KillMode = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i KillSignal = ...;
```

```
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i RestartKillSignal = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i FinalKillSignal = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b SendSIGKILL = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b SendSIGHUP = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i WatchdogSignal = ...;
    };
    interface org.freedesktop.DBus.Peer { ... };
    interface org.freedesktop.DBus.Introspectable { ... };
    interface org.freedesktop.DBus.Properties { ... };
    interface org.freedesktop.systemd1.Unit { ... };
};
```

Properties

Most of the properties map directly to the corresponding settings in mount unit files. As mount units invoke the /usr/bin/mount command, their bus objects include implicit *ExecMount* (and similar) fields which contain information about processes to execute. They also share most of the fields related to the execution context that Service objects expose (see above). In addition to these properties there are the following:

ControlPID contains the PID of the currently running /usr/bin/mount or /usr/bin/umount command if there is one running, otherwise 0.

Result contains a value explaining why a mount unit failed if it failed. It can take the values "success", "resources", "timeout", "exit−code", "signal", or "core−dump" which have the identical meaning as the corresponding values of the corresponding field of service unit objects (see above).

AUTOMOUNT UNIT OBJECTS

All automount unit objects implement the org.freedesktop.systemd1.Automount interface (described here) in addition to the generic org.freedesktop.systemd1.Unit interface (see above).

```
node /org/freedesktop/systemd1/unit/proc_2dsys_2dfs_2dbinfmt_5fmisc_2eautomount {
  interface org.freedesktop.systemd1.Automount {
    properties:
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly s Where = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly u DirectoryMode = ...;
      readonly s Result = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly t TimeoutIdleUSec = ...;
  };
  interface org.freedesktop.DBus.Peer { ... };
  interface org.freedesktop.DBus.Introspectable { ... };
  interface org.freedesktop.DBus.Properties { ... };
  interface org.freedesktop.systemd1.Unit { ... };
};
```

Properties

Most of the properties map directly to the corresponding settings in the automount unit files.

Result knows the values "success" and "resources" at this time. They have the same meanings as the corresponding values of the corresponding field of the Service object.

TIMER UNIT OBJECTS

All timer unit objects implement the org.freedesktop.systemd1.Timer interface (described here) in addition to the generic org.freedesktop.systemd1.Unit interface (see above).

```
node /org/freedesktop/systemd1/unit/systemd_2dtmpfiles_2dclean_2etimer {
  interface org.freedesktop.systemd1.Timer {
    properties:
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly s Unit = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
```

```
        readonly a(stt) TimersMonotonic = [...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
        readonly a(sst) TimersCalendar = [...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b OnClockChange = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b OnTimezoneChange = ...;
        readonly t NextElapseUSecRealtime = ...;
        readonly t NextElapseUSecMonotonic = ...;
        readonly t LastTriggerUSec = ...;
        readonly t LastTriggerUSecMonotonic = ...;
        readonly s Result = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t AccuracyUSec = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly t RandomizedDelayUSec = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b FixedRandomDelay = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b Persistent = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b WakeSystem = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b RemainAfterElapse = ...;
      };
      interface org.freedesktop.DBus.Peer { ... };
      interface org.freedesktop.DBus.Introspectable { ... };
      interface org.freedesktop.DBus.Properties { ... };
      interface org.freedesktop.systemd1.Unit { ... };
    };
```

Properties

Unit contains the name of the unit to activate when the timer elapses.

TimersMonotonic contains an array of structs that contain information about all monotonic timers of this timer unit. The structs contain a string identifying the timer base, which is one of "OnActiveUSec",

"OnBootUSec", "OnStartupUSec", "OnUnitActiveUSec", or "OnUnitInactiveUSec" which correspond to the settings of the same names in the timer unit files; the microsecond offset from this timer base in monotonic time; the next elapsation point on the **CLOCK_MONOTONIC** clock, relative to its epoch.

TimersCalendar contains an array of structs that contain information about all realtime/calendar timers of this timer unit. The structs contain a string identifying the timer base, which may only be "OnCalendar" for now; the calendar specification string; the next elapsation point on the **CLOCK_REALTIME** clock, relative to its epoch.

NextElapseUSecRealtime contains the next elapsation point on the **CLOCK_REALTIME** clock in miscroseconds since the epoch, or 0 if this timer event does not include at least one calendar event.

Similarly, *NextElapseUSecMonotonic* contains the next elapsation point on the **CLOCK_MONOTONIC** clock in microseconds since the epoch, or 0 if this timer event does not include at least one monotonic event.

Result knows the values "success" and "resources" with the same meanings as the matching values of the corresponding property of the service interface.

SWAP UNIT OBJECTS

All swap unit objects implement the org.freedesktop.systemd1.Swap interface (described here) in addition to the generic org.freedesktop.systemd1.Unit interface (see above).

```
node /org/freedesktop/systemd1/unit/dev_2dsda3_2eswap {
  interface org.freedesktop.systemd1.Swap {
    methods:
      GetProcesses(out a(sus) processes);
      AttachProcesses(in  s subcgroup,
                      in  au pids);
    properties:
      readonly s What = '...';
      readonly i Priority = ...;
      readonly s Options = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly t TimeoutUSec = ...;
      readonly u ControlPID = ...;
      readonly s Result = '...';
      readonly u UID = ...;
      readonly u GID = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
      readonly a(sasbttttuii) ExecActivate = [...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("invalidates")
      readonly a(sasbttttuii) ExecDeactivate = [...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly s Slice = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly s ControlGroup = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t MemoryCurrent = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t MemoryAvailable = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t CPUUsageNSec = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly ay EffectiveCPUs = [...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly ay EffectiveMemoryNodes = [...];
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t TasksCurrent = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPIngressBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPIngressPackets = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPEgressBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IPEgressPackets = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOReadBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOReadOperations = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOWriteBytes = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOWriteOperations = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b Delegate = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as DelegateControllers = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b CPUAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupCPUWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUShares = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupCPUShares = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUQuotaPerSecUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUQuotaPeriodUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay AllowedCPUs = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay AllowedMemoryNodes = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b IOAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IODeviceWeight = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOReadBandwidthMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOWriteBandwidthMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOReadIOPSMax = [...];
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOWriteIOPSMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IODeviceLatencyTargetUSec = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b BlockIOAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t BlockIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupBlockIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIODeviceWeight = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIOReadBandwidth = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIOWriteBandwidth = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b MemoryAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t DefaultMemoryLow = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t DefaultMemoryMin = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryMin = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryLow = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryHigh = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemorySwapMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryLimit = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s DevicePolicy = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(ss) DeviceAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b TasksAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t TasksMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b IPAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iayu) IPAddressAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iayu) IPAddressDeny = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as IPIngressFilterPath = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as IPEgressFilterPath = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as DisableControllers = ['...', ...];
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMSwap = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMMemoryPressure = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly u ManagedOOMMemoryPressureLimit = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMPreference = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(ss) BPFProgram = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iiqq) SocketBindAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iiqq) SocketBindDeny = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as Environment = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(sb) EnvironmentFiles = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as PassEnvironment = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as UnsetEnvironment = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u UMask = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCPU = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCPUSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitFSIZE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitFSIZESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitDATA = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitDATASoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSTACK = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSTACKSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCORE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitCORESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRSS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRSSSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNOFILE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNOFILESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitAS = ...;
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitASSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNPROC = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNPROCSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMEMLOCK = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMEMLOCKSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitLOCKS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitLOCKSSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSIGPENDING = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitSIGPENDINGSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMSGQUEUE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitMSGQUEUESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNICE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitNICESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTPRIO = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTPRIOSoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTTIME = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LimitRTTIMESoft = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s WorkingDirectory = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootDirectory = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootImage = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ss) RootImageOptions = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay RootHash = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootHashPath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay RootHashSignature = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootHashSignaturePath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RootVerity = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(sba(ss)) ExtensionImages = [...];
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ssba(ss)) MountImages = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i OOMScoreAdjust = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t CoredumpFilter = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i Nice = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i IOSchedulingClass = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i IOSchedulingPriority = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i CPUSchedulingPolicy = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i CPUSchedulingPriority = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay CPUAffinity = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b CPUAffinityFromNUMA = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i NUMAPolicy = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay NUMAMask = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t TimerSlackNSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b CPUSchedulingResetOnFork = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b NonBlocking = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardInput = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardInputFileDescriptorName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly ay StandardInputData = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardOutput = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardOutputFileDescriptorName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardError = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s StandardErrorFileDescriptorName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s TTYPath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b TTYReset = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b TTYVHangup = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b TTYVTDisallocate = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SyslogPriority = ...;
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s SyslogIdentifier = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b SyslogLevelPrefix = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SyslogLevel = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SyslogFacility = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i LogLevelMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t LogRateLimitIntervalUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u LogRateLimitBurst = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly aay LogExtraFields = [[...], ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s LogNamespace = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SecureBits = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t CapabilityBoundingSet = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t AmbientCapabilities = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s User = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s Group = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b DynamicUser = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b RemoveIPC = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(say) SetCredential = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ss) LoadCredential = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as SupplementaryGroups = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s PAMName = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ReadWritePaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ReadOnlyPaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as InaccessiblePaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ExecPaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as NoExecPaths = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t MountFlags = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateTmp = ...;
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateDevices = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectClock = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectKernelTunables = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectKernelModules = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectKernelLogs = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectControlGroups = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateNetwork = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateUsers = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateMounts = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b PrivateIPC = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s ProtectHome = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s ProtectSystem = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b SameProcessGroup = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s UtmpIdentifier = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s UtmpMode = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bs) SELinuxContext = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bs) AppArmorProfile = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bs) SmackProcessLabel = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b IgnoreSIGPIPE = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b NoNewPrivileges = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bas) SystemCallFilter = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as SystemCallArchitectures = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i SystemCallErrorNumber = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bas) SystemCallLog = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s Personality = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b LockPersonality = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly (bas) RestrictAddressFamilies = ...;
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s RuntimeDirectoryPreserve = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u RuntimeDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as RuntimeDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u StateDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as StateDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u CacheDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as CacheDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u LogsDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as LogsDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly u ConfigurationDirectoryMode = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly as ConfigurationDirectory = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t TimeoutCleanUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b MemoryDenyWriteExecute = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b RestrictRealtime = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b RestrictSUIDSGID = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly t RestrictNamespaces = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ssbt) BindPaths = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ssbt) BindReadOnlyPaths = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly a(ss) TemporaryFileSystem = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b MountAPIVFS = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s KeyringMode = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s ProtectProc = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s ProcSubset = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly b ProtectHostname = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s NetworkNamespacePath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s IPCNamespacePath = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s KillMode = '...';
```

```
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i KillSignal = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i RestartKillSignal = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i FinalKillSignal = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b SendSIGKILL = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b SendSIGHUP = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i WatchdogSignal = ...;
    };
    interface org.freedesktop.DBus.Peer { ... };
    interface org.freedesktop.DBus.Introspectable { ... };
    interface org.freedesktop.DBus.Properties { ... };
    interface org.freedesktop.systemd1.Unit { ... };
};
```

Properties

Most of the properties map directly to the corresponding settings in swap unit files. As mount units invoke the **swapon**(8) command, their bus objects include implicit *ExecActivate* (and similar) fields which contain information about processes to execute. They also share most of the fields related to the execution context that Service objects expose (see above). In addition to these properties there are the following:

ControlPID contains the PID of the currently running **swapon**(8) or **swapoff**(8) command if there is one running, otherwise 0.

Result contains a value explaining why a mount unit failed if it failed. It can take the values "success", "resources", "timeout", "exit−code", "signal", or "core−dump" which have the identical meanings as the corresponding values of the corresponding field of service unit objects (see above).

PATH UNIT OBJECTS

```
node /org/freedesktop/systemd1/unit/cups_2epath {
  interface org.freedesktop.systemd1.Path {
    properties:
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly s Unit = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly a(ss) Paths = [...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly b MakeDirectory = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly u DirectoryMode = ...;
      readonly s Result = '...';
  };
  interface org.freedesktop.DBus.Peer { ... };
  interface org.freedesktop.DBus.Introspectable { ... };
  interface org.freedesktop.DBus.Properties { ... };
  interface org.freedesktop.systemd1.Unit { ... };
};
```

Properties

Most properties correspond directly with the matching settings in path unit files.

The others:

Paths contains an array of structs. Each struct contains the condition to watch, which can be one of "PathExists", "PathExistsGlob", "PathChanged", "PathModified", or "DirectoryNotEmpty" which correspond directly to the matching settings in the path unit files; and the path to watch, possibly including glob expressions.

Result contains a result value which can be "success" or "resources" which have the same meaning as the corresponding field of the Service interface.

SLICE UNIT OBJECTS

All slice unit objects implement the org.freedesktop.systemd1.Slice interface (described here) in addition to the generic org.freedesktop.systemd1.Unit interface (see above).

```
node /org/freedesktop/systemd1/unit/system_2eslice {
  interface org.freedesktop.systemd1.Slice {
    methods:
      GetProcesses(out a(sus) processes);
      AttachProcesses(in  s subcgroup,
                      in  au pids);
    properties:
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly s Slice = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly s ControlGroup = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t MemoryCurrent = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t MemoryAvailable = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t CPUUsageNSec = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly ay EffectiveCPUs = [...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly ay EffectiveMemoryNodes = [...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t TasksCurrent = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IPIngressBytes = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IPIngressPackets = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IPEgressBytes = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IPEgressPackets = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IOReadBytes = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IOReadOperations = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IOWriteBytes = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IOWriteOperations = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly b Delegate = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly as DelegateControllers = ['...', ...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly b CPUAccounting = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t CPUWeight = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t StartupCPUWeight = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t CPUShares = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t StartupCPUShares = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
```

```
readonly t CPUQuotaPerSecUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUQuotaPeriodUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay AllowedCPUs = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay AllowedMemoryNodes = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b IOAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IODeviceWeight = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOReadBandwidthMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOWriteBandwidthMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOReadIOPSMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOWriteIOPSMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IODeviceLatencyTargetUSec = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b BlockIOAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t BlockIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupBlockIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIODeviceWeight = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIOReadBandwidth = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIOWriteBandwidth = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b MemoryAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t DefaultMemoryLow = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t DefaultMemoryMin = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryMin = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryLow = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryHigh = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemorySwapMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
```

```
        readonly t MemoryLimit = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly s DevicePolicy = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly a(ss) DeviceAllow = [...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly b TasksAccounting = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly t TasksMax = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly b IPAccounting = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly a(iayu) IPAddressAllow = [...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly a(iayu) IPAddressDeny = [...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly as IPIngressFilterPath = ['...', ...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly as IPEgressFilterPath = ['...', ...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly as DisableControllers = ['...', ...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly s ManagedOOMSwap = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly s ManagedOOMMemoryPressure = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly u ManagedOOMMemoryPressureLimit = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly s ManagedOOMPreference = '...';
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly a(ss) BPFProgram = [...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly a(iiqq) SocketBindAllow = [...];
        @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
        readonly a(iiqq) SocketBindDeny = [...];
    };
    interface org.freedesktop.DBus.Peer { ... };
    interface org.freedesktop.DBus.Introspectable { ... };
    interface org.freedesktop.DBus.Properties { ... };
    interface org.freedesktop.systemd1.Unit { ... };
};
```

Properties

Most properties correspond directly with the matching settings in slice unit files.

SCOPE UNIT OBJECTS

All scope unit objects implement the org.freedesktop.systemd1.Scope interface (described here) in addition to the generic org.freedesktop.systemd1.Unit interface (see above).

```
node /org/freedesktop/systemd1/unit/session_2d1_2escope {
  interface org.freedesktop.systemd1.Scope {
    methods:
      Abandon();
      GetProcesses(out a(sus) processes);
      AttachProcesses(in  s subcgroup,
                      in  au pids);
    signals:
      RequestStop();
    properties:
      readonly s Controller = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly t TimeoutStopUSec = ...;
      readonly s Result = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly t RuntimeMaxUSec = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly s Slice = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly s ControlGroup = '...';
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t MemoryCurrent = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t MemoryAvailable = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t CPUUsageNSec = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly ay EffectiveCPUs = [...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly ay EffectiveMemoryNodes = [...];
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t TasksCurrent = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IPIngressBytes = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IPIngressPackets = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IPEgressBytes = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IPEgressPackets = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IOReadBytes = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IOReadOperations = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IOWriteBytes = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t IOWriteOperations = ...;
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b Delegate = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as DelegateControllers = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b CPUAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupCPUWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUShares = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupCPUShares = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUQuotaPerSecUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t CPUQuotaPeriodUSec = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay AllowedCPUs = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly ay AllowedMemoryNodes = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b IOAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t IOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IODeviceWeight = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOReadBandwidthMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOWriteBandwidthMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOReadIOPSMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IOWriteIOPSMax = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) IODeviceLatencyTargetUSec = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b BlockIOAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t BlockIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t StartupBlockIOWeight = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIODeviceWeight = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIOReadBandwidth = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(st) BlockIOWriteBandwidth = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b MemoryAccounting = ...;
```

```
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t DefaultMemoryLow = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t DefaultMemoryMin = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryMin = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryLow = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryHigh = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemorySwapMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t MemoryLimit = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s DevicePolicy = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(ss) DeviceAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b TasksAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly t TasksMax = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly b IPAccounting = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iayu) IPAddressAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iayu) IPAddressDeny = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as IPIngressFilterPath = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as IPEgressFilterPath = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly as DisableControllers = ['...', ...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMSwap = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMMemoryPressure = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly u ManagedOOMMemoryPressureLimit = ...;
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly s ManagedOOMPreference = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(ss) BPFProgram = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iiqq) SocketBindAllow = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("false")
readonly a(iiqq) SocketBindDeny = [...];
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly s KillMode = '...';
@org.freedesktop.DBus.Property.EmitsChangedSignal("const")
readonly i KillSignal = ...;
```

```
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i RestartKillSignal = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i FinalKillSignal = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b SendSIGKILL = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly b SendSIGHUP = ...;
        @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
        readonly i WatchdogSignal = ...;
    };
    interface org.freedesktop.DBus.Peer { ... };
    interface org.freedesktop.DBus.Introspectable { ... };
    interface org.freedesktop.DBus.Properties { ... };
    interface org.freedesktop.systemd1.Unit { ... };
};
```

Methods

Abandon() may be used to place a scope unit in the "abandoned" state. This may be used to inform the system manager that the manager that created the scope lost interest in the scope (for example, because it is terminating), without wanting to shut down the scope entirely.

Signals

RequestStop is sent to the peer that is configured in the *Controller* property when systemd is requested to terminate the scope unit. A program registering a scope can use this to cleanly shut down the processes it added to the scope instead of letting systemd do it with the usual **SIGTERM** logic.

Properties

All properties correspond directly with the matching properties of service units.

Controller contains the bus name (unique or well−known) that is notified when the scope unit is to be shut down via a **RequestStop** signal (see below). This is set when the scope is created. If not set, the scope's processes will terminated with **SIGTERM** directly.

JOB OBJECTS

Job objects encapsulate scheduled or running jobs. Each unit can have none or one jobs in the execution queue. Each job is attached to exactly one unit.

```
node /org/freedesktop/systemd1/job/666 {
  interface org.freedesktop.systemd1.Job {
    methods:
      Cancel();
      GetAfter(out a(usssoo) jobs);
      GetBefore(out a(usssoo) jobs);
    properties:
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly u Id = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly (so) Unit = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("const")
      readonly s JobType = '...';
      readonly s State = '...';
  };
  interface org.freedesktop.DBus.Peer { ... };
  interface org.freedesktop.DBus.Introspectable { ... };
  interface org.freedesktop.DBus.Properties { ... };
};
```

Methods

Cancel() cancels the job. Note that this will remove a job from the queue if it is not yet executed but generally will not cause a job that is already in the process of being executed to be aborted. This operation may also be requested via the **CancelJob()** method of the Manager object (see above), which is sometimes useful to reduce roundtrips.

Properties

Id is the numeric Id of the job. During the runtime of a systemd instance each numeric ID is only assigned once.

Unit refers to the unit this job belongs to. It is a structure consisting of the name of the unit and a bus path to the unit's object.

JobType refers to the job's type and is one of "start", "verify−active", "stop", "reload", "restart", "try−restart", or "reload−or−start". Note that later versions might define additional values.

State refers to the job's state and is one of "waiting" and "running". The former indicates that a job is currently queued but has not begun to execute yet. The latter indicates that a job is currently being executed.

EXAMPLES

Example 1. Introspect org.freedesktop.systemd1.Manager on the bus

```
$ gdbus introspect −−system \
      −−dest org.freedesktop.systemd1 \
      −−object−path /org/freedesktop/systemd1
```

Example 2. Introspect a unit on the bus

```
$ busctl introspect org.freedesktop.systemd1 \
  $(busctl call org.freedesktop.systemd1 \
    /org/freedesktop/systemd1 \
    org.freedesktop.systemd1.Manager \
    GetUnit s systemd−resolved.service | cut −d"" −f2)
```

Example 3. Introspect org.freedesktop.systemd1.Job on the bus

```
$ gdbus introspect −−system −−dest org.freedesktop.systemd1 \
  −−object−path /org/freedesktop/systemd1/job/1292
```

VERSIONING

These D−Bus interfaces follow **the usual interface versioning guidelines**[4].

NOTES

1. polkit
   https://www.freedesktop.org/software/polkit/docs/latest/
2. New Control Group Interface
   https://www.freedesktop.org/wiki/Software/systemd/ControlGroupInterface/
3. Booting Without /usr is Broken
   http://freedesktop.org/wiki/Software/systemd/separate-usr-is-broken
4. the usual interface versioning guidelines
   http://0pointer.de/blog/projects/versioning-dbus.html