NAME

org.freedesktop.timedate1 − The D−Bus interface of systemd−timedated

INTRODUCTION

systemd-timedated.service(8) is a system service that can be used to control the system time and related settings. This page describes the D−Bus interface.

THE D−BUS API

The service exposes the following interfaces on the bus:

```
node /org/freedesktop/timedate1 {
  interface org.freedesktop.timedate1 {
    methods:
      SetTime(in  x usec_utc,
              in  b relative,
              in  b interactive);
      SetTimezone(in  s timezone,
                  in  b interactive);
      SetLocalRTC(in  b local_rtc,
                  in  b fix_system,
                  in  b interactive);
      SetNTP(in  b use_ntp,
             in  b interactive);
      ListTimezones(out as timezones);
    properties:
      readonly s Timezone = '...';
      readonly b LocalRTC = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly b CanNTP = ...;
      readonly b NTP = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly b NTPSynchronized = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t TimeUSec = ...;
      @org.freedesktop.DBus.Property.EmitsChangedSignal("false")
      readonly t RTCTimeUSec = ...;
  };
  interface org.freedesktop.DBus.Peer { ... };
  interface org.freedesktop.DBus.Introspectable { ... };
  interface org.freedesktop.DBus.Properties { ... };
};
```

Methods

Use **SetTime()** to change the system clock. Pass a value of microseconds since the UNIX epoch (1 Jan 1970 UTC). If *relative* is true, the passed usec value will be added to the current system time. If it is false, the current system time will be set to the passed usec value. If the system time is set with this method, the RTC will be updated as well.

Use **SetTimezone()** to set the system timezone. Pass a value like "Europe/Berlin" to set the timezone. Valid timezones are listed in /usr/share/zoneinfo/zone.tab. If the RTC is configured to be maintained in local time, it will be updated accordingly.

Use **SetLocalRTC()** to control whether the RTC is in local time or UTC. It is strongly recommended to maintain the RTC in UTC. However, some OSes (Windows) maintain the RTC in local time, which might make it necessary to enable this feature. Note that this might create various problems as daylight changes could be missed. If *fix_system* is "true", the time from the RTC is read again and the system clock is adjusted according to the new setting. If *fix_system* is "false", the system time is written to the RTC taking the new setting into account. Use *fix_system=true* in installers and livecds where the RTC is probably more reliable than the system time. Use *fix_system=false* in configuration UIs that are run during normal operation and where the system clock is probably more reliable than the RTC.

Use **SetNTP()** to control whether the system clock is synchronized with the network using systemd−timesyncd. This will enable and start or disable and stop the chosen time synchronization service.

ListTimezones() returns a list of time zones known on the local system as an array of names ("["Africa/Abidjan", "Africa/Accra", ..., "UTC"]").

Properties

Timezone shows the currently configured time zone. *LocalRTC* shows whether the RTC is configured to use UTC (false), or the local time zone (true). *CanNTP* shows whether a service to perform time synchronization over the network is available, and *NTP* shows whether such a service is enabled.

NTPSynchronized shows whether the kernel reports the time as synchronized (c.f. **adjtimex**(3)). *TimeUSec* and *RTCTimeUSec* show the current time on the system and in the RTC. The purpose of those three properties is to allow remote clients to access this information over D−Bus. Local clients can access the information directly.

Whenever the *Timezone* and *LocalRTC* settings are changed via the daemon, **PropertyChanged** signals are sent out to which clients can subscribe.

Note that this service will not inform you about system time changes. Use **timerfd**(3) with **CLOCK_REALTIME** and **TFD_TIMER_CANCEL_ON_SET** for that.

Security

The *interactive* boolean parameters can be used to control whether **polkit**[1] should interactively ask the user for authentication credentials if required.

The polkit action for **SetTimezone()** is org.freedesktop.timedate1.set−timezone. For **SetLocalRTC()** it is org.freedesktop.timedate1.set−local−rtc, for **SetTime()** it is org.freedesktop.timedate1.set−time and for **SetNTP()** it is org.freedesktop.timedate1.set−ntp. **ListTimezones()** does not require any privileges.

EXAMPLES

Example 1. Introspect org.freedesktop.timedate1 on the bus

```
$ gdbus introspect −−system \
  −−dest org.freedesktop.timedate1 \
  −−object−path /org/freedesktop/timedate1
```

VERSIONING

These D−Bus interfaces follow **the usual interface versioning guidelines**[2].

SEE ALSO

More information on how the system clock and RTC interact[3]

NOTES

1. polkit
   https://www.freedesktop.org/software/polkit/docs/latest/
2. the usual interface versioning guidelines
   http://0pointer.de/blog/projects/versioning-dbus.html
3. More information on how the system clock and RTC interact
   https://lists.freedesktop.org/archives/systemd-devel/2011-May/002526.html