24.9.0 D-Bus

Diese Einführung in das Thema D-Bus subsumiert Teile aus unterschiedlichen Literatur-Quellen, um erste Antworten auf die Fragen "Was ist D-Bus?" und "Was kann D-Bus leisten?" zu geben.

D-Bus oder Desktop-Bus ist ein Framework der Inter-Process-Communication (IPC) und Teil des free-desktop-Projektes (https://www.freedesktop.org/wiki/). IPC gestattet es Prozessen, Daten in Form von Nachrichten auszutauschen. Diese Nachrichten werden auf einem Nachrichtenkanal gesendet und empfangen. Diesen Kanal nennt man D-Bus. D-Bus bietet mindestens zwei Busse zur Kommunikation zwischen Prozessen an. Einer ist der globale System-Bus und ein anderer der Session-Bus, zu dem jeder Desktop eines angemeldeten Benutzers innerhalb seiner Desktop-Sitzung automatisch verbunden wird, wenn sich eine seiner Anwendungen am D-Bus-Bus registriert.

Nach dem Einschalten eines Computers mit einem Linux-Betriebssystem werden auch Programme gestartet, die sich zum System-Bus verbinden und sich dort registrieren. Wenn sich ein Benutzer am Desktop angemeldet hat, dann starten weitere Programme und melden sich am Session-Bus an. Gemeinsam ist allen am D-Bus registrierten Programmen, dass sie entweder einen bestimmten Nachrichten-Dienst anbieten oder angebotene Nachrichten-Dienste nutzen. Ein Programm, das sich am System-Bus oder am Session-Bus registrieren kann, soll als dbus-fähig gekennzeichnet werden.

Beispiel: Wenn Sie in Mint 18.3 in der Taskleiste auf das Netzwerksymbol klicken, dann öffnet sich ein kleines Menü-Fenster, in dem Sie das LAN oder WLAN aus- oder einschalten können. Nach dem Ausschalten sendet der Dienst mit der ID :1.21 – hinter dem der D-Bus-Dienst org.gnome.networkmanager_applet steht – u.a. das Signal 'NewIcon'. Der Dienst org.gnome.freedesktop.Notifikations empfängt es und wertet das Signal aus. Als erste Reaktion ändert sich das Netzwerksymbol in der Taskleiste. Dann schickt der Dienst eine Nachricht mit dem Namen 'Notify' zum D-Bus. Der Inhalt der Nachricht ist eine Mitteilung und wird in einem Fenster auf dem Desktop angezeigt:

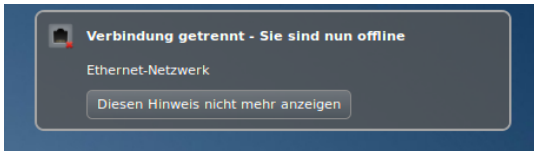

Abbildung 24.9.0.1: Nachricht auf dem Desktop

Jetzt werden Sie sich sicher fragen

- woher die Informationen aus dem letzten Abschnitt stammen,
- welche weiteren Nachrichten der o.a. Dienst senden kann und
- ob Sie als Gambas-Programmierer angebotene Dienste auf dem D-Bus nutzen können oder mit d-bus-fähigen Gambas-Programmen eigene Dienste auf dem D-Bus anbieten können?

1. Antwort
Die Antwort ist einfach: Um den Datenverkehr auf dem System-Bus oder auf dem Session-Bus zu beobachten, können Sie wie der Autor neben Andern das Konsolen-Programm 'dbus-monitor' einsetzen.

Mit den folgenden Befehlen können Sie in einer Konsole eine Liste der Anwendungen auslesen, die am System-Bus und am Session-Bus registriert sind. Die Liste enthält die D-Bus-Namen und die ihnen zugeordneten einzigartigen Verbindungs-IDs, denen stets ein Doppelpunkt vorangestellt wird. Eingesetzt wird das Programm *qdbus*:

```
$ qdbus --session | grep -v ":"
  org.gnome.SessionManager
  org.gtk.Private.UDisks2VolumeMonitor
  ...
  org.PulseAudio1
  org.freedesktop.DBus
```

```
$ qdbus --system | grep ":"
...
:1.44
:1.5
:1.21
```

Sie können aber auch das Gambas-Programm 'DBusView' einsetzen und erhalten diese Übersicht zu allen aktuell auf dem D-Bus registrierten Programmen:

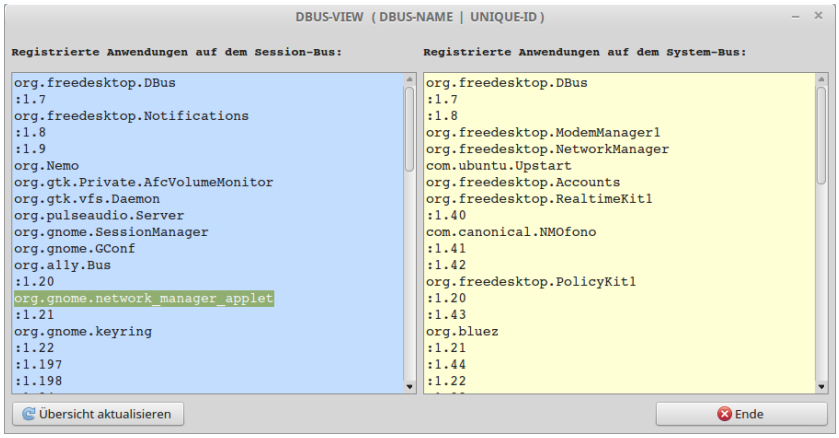

Abbildung 24.9.0.2: Registrierte Programme auf dem D-Bus

2. Antwort
Sie müssen wissen, das jedes am D-Bus registrierte Programm in einer XML-Datei eine Beschreibung des angebotenen Dienstes zur Verfügung stellt. Es gibt mehrere Wege, um sich den Inhalt der XML-Datei anzusehen. Kennen Sie bereits den D-Bus-Namen der Anwendung – zum Beispiel über das Programm 'DBusView' – dann besteht ein Weg im Aufruf der Methode 'Introspect' der Standard-Schnittstelle 'org.freedesktop.DBus.Introspectable'. Diesen Aufruf nennt man Introspection. Da der Introspection eine zentrale Bedeutung bei der Arbeit mit den Diensten auf dem D-Bus zukommt, erfolgt ihre Beschreibung in einem eigenen Kapitel 24.9.0.2. Um sich über alle Programme und deren Dienste auf dem System-Bus oder auf dem Session-Bus zu informieren, ist das Programm 'd-feet' die erste Wahl, dessen Verwendung im Kapitel 24.9.0.2.3 ausführlich beschrieben wird.

3. Antwort
Ja – die Komponente gb.dbus stellt Ihnen mehrere Klassen zur Verfügung, um Dienste auf dem D-Bus zu nutzen oder um mit eigenen d-bus-fähigen Gambas-Programmen Dienste anzubieten oder Signale abzufangen oder zu senden.

In den folgenden Kapiteln werden diese Klassen mit ihren Eigenschaften, Methoden und Ereignissen beschrieben. Ergänzend werden Ihnen zahlreiche Projekte vorgestellt, die Anregungen für eigene d-bus-fähige Programme geben sollen.

Übersicht der Kapitel

- Kapitel 24.9.0.1 Gambas und DBus: Im Mittelpunkt stehen D-Bus, D-Bus-Nachricht, D-Bus-Name, D-Bus-Adresse, D-Bus-Objekt und D-Bus-Interface.
- Kapitel 24.9.0.2 D-Bus-Introspection: Die Fähigkeit eines Programms, die Eigenschaften eines exportierten D-Bus-Objekts zur Laufzeit zu untersuchen wird als *Introspection* bezeichnet. Gambas besitzt mit der Komponente *gb.dbus* die Fähigkeit einen Dienst, dessen Objekte, Schnittstellen sowie deren Methoden und Eigenschaften aber auch deklarierte Signale auf dem D-Bus zu analysieren und in geeigneter Form abzubilden. Es werden viele Programme vorgestellt, mit denen spezifische Aspekte analysiert werden können.
- Kapitel 24.9.0.3 D-Bus-Signatur: Sie benötigen Signaturen, um die D-Bus-Daten mit ihren D-Bus-Datentypen auf Gambas-Daten und deren Datentypen abzubilden und umgekehrt. Zwei umfangreiche Beispiele mit komplexen Signaturen zeigen die notwendige Daten-Konvertierung.
- Kapitel 24.9.1 Klasse DBusObject: Die (statische) Klasse ist die Elternklasse für alle Objekte, die Sie zum D-Bus exportieren können.
- Kapitel 24.9.2 Klasse DBus: Die Klasse DBus verwaltet die Verbindung einer Anwendung zum Session-Bus oder zum System-Bus. Mit den Methoden können Sie ein D-Bus-Objekt am D-Bus registrieren, de-registrieren oder prüfen, ob eine d-bus-fähige Anwendung am D-Bus registriert ist. Mit Hilfe der Raise(...)-Methode können Sie ein D-Bus-Signal senden. Um zu prüfen, ob ein spezielles D-Bus-Objekt Ihrer Gambas-Anwendung bereits am D-Bus registriert ist, setzen Sie die IsRegistered(..)-Methode ein.
- Kapitel 24.9.3 Klasse DBusApplication: Die Klasse *DBusApplication* (gb.dbus) repräsentiert eine

Anwendung, die am D-Bus registriert ist und nutzt die Klasse DBusConnection.

- Kapitel 24.9.4 Klasse DBusConnection: Die Klasse repräsentiert eine Verbindung einer d-bus-fähigen Anwendung zum System-D-Bus oder zum Session-D-Bus. Sie wird vorrangig dazu eingesetzt, um die auf dem System-D-Bus registrierten Anwendungen zu ermitteln.
- Kapitel 24.9.5 Klasse DBusObserver: In diesem Kapitel erfahren Sie, wie Sie eine Nachricht auf dem D-Bus abfangen und deren Inhalt anzeigen. Zwei Projekte demonstrieren den Einsatz der Klasse.
- Kapitel 24.9.6 Klasse DBusSignal: Wenn Sie nur Signale abfangen und deren Inhalt anzeigen wollen, dann stellt diese Klasse einen speziellen Observer zu Verfügung. Drei Projekte ergänzen die Theorie.
- Kapitel 24.9.7 Klasse DBusProxy: Diese Klasse zeigt, wie Sie einen D-Bus-Proxy definieren und einsetzen.
- Kapitel 24.9.8 Klasse DBusVariant und DBusValues: Mit Hilfe der beiden Klassen können Sie Methoden und Signale definieren und so mit einem Gambas-Programm Dienste auf dem D-Bus anbieten und Signale senden. Die Projekte sind immer als Server-Client-Projekte angelegt.

Die folgende Link-Liste führt Sie auf interessante Webseiten, die sich jeweils in spezieller Weise dem Thema D-Bus zuwenden:

- https://techbase.kde.org/Development/Tutorials/D-Bus/Introduction_(de)
- https://www.freedesktop.org/wiki/Software/dbus/
- http://www.linuxjournal.com/article/10455?page=0,0
- https://blog.mafr.de/2006/12/19/using-dbus-introspection/
- https://pythonhosted.org/txdbus/dbus_overview.html
- http://gambaswiki.org/wiki/comp/gb.dbus
- http://gambaswiki.org/wiki/doc/dbus