#### 24.9.0.2 Introspection

Die Fähigkeit eines Programms, die Eigenschaften eines exportierten D-Bus-Objektes 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.

Wenn Sie die Inter-Process-Communication (IPC) über den D-Bus in Ihren Gambas-Programmen einsetzen wollen, dann ist es gut zu wissen:

- welche Anwendungen aktuell auf dem System-D-Bus und auf dem Session-D-Bus registriert sind, denn nicht jedes Programm ist dbus-fähig und kann sich in den D-Bus einklinken,
- welche Adressen für diese Anwendungen benutzt werden. Das sichert die Kenntnis über den D-Bus-Namen und den Objekt-Pfad sowie optional zu den Schnittstellen (Methoden, Eigenschaften und Ereignisse), sofern diese Schnittstellen deklariert sind,
- ob eine Beschreibung der Schnittstellen in einem XML-Dokument mit einer speziellen doctype-Deklaration existiert und wenn ja, welchen Inhalt dieses Dokument hat.

##### 24.9.0.2.1 Introspection Konsole

Die Standard-Schnittstelle *org.freedesktop.DBus.Introspectable* besitzt zum Beispiel die Methode *Introspect()*, mit der Sie eine existierende Schnittstellenbeschreibung in ein XML-Dokument einlesen können. Für den d-bus-fähigen Netzwerk-Manager zum Beispiel rufen Sie diese beiden Befehle auf:

```
$ dbus-send --system --type=method_call --print-reply --dest=org.freedesktop.NetworkManager \
 /org/freedesktop/NetworkManager  org.freedesktop.DBus.Introspectable.Introspect > nwm.xml
```

```
$ xed $HOME/nwm.xml
```

Das ist der stark gekürzte Inhalt der D-Bus-Schnittstellen-Beschreibung des Netzwerk-Managers:

```
method return sender=:1.5 -> dest=:1.71 reply_serial=2
   string "<!DOCTYPE node PUBLIC "-//freedesktop//DTD D-BUS Object Introspection 1.0//EN"
"http://www.freedesktop.org/standards/dbus/1.0/introspect.dtd">
<node>
  <interface name="org.freedesktop.DBus.Introspectable">
    <method name="Introspect">
      <arg name="data" direction="out" type="s"/>
    </method>
  </interface>
  <interface name="org.freedesktop.DBus.Properties">
    <method name="Get">
      <arg name="interface" direction="in" type="s"/>
      <arg name="propname" direction="in" type="s"/>
      <arg name="value" direction="out" type="v"/>
    </method>
...
    <method name="GetAll">
      <arg name="interface" direction="in" type="s"/>
      <arg name="props" direction="out" type="a{sv}"/> ' type steht für Datentyp
    </method>
  </interface>
...
  <node name="DHCP4Config"/>
  <node name="IP6Config"/>
  <node name="Settings"/>

</node>
```

In den nächsten vier Abschnitten werden weitere Einsatzfälle für die Konsolen-Programme dbus-send, qdbus und dbus-monitor vorgestellt:

```
$ qdbus --help
Usage: qdbus [--system | --address ADDRESS] [--literal] [servicename] [path] [method] [args]
```

(1)
Ausgabe der Liste aller D-Bus-Namen von Anwendungen (Diensten) auf dem System-Bus:

```
$ qdbus --system | grep -v ':'
...
 org.freedesktop.ConsoleKit
```

```
...
 org.freedesktop.NetworkManager
 org.freedesktop.PolicyKit1
 org.freedesktop.DBus
```

Alternativ können Sie auch mit dem Programm *dbus-send* arbeiten:

```
SYNTAX: $ dbus-send --session --print-reply –dest=DienstName /Objekt-Pfad Schnittstelle
```

```
$ dbus-send --session --print-reply --dest=org.freedesktop.DBus /org/freedesktop/DBus \
  org.freedesktop.DBus.ListActivatableNames
```

(2)
So erhalten Sie die Liste aller Schnittstellen (Interfaces) zu einem *ausgewählten Dienst* aus (1):

```
$ qdbus --system org.freedesktop.NetworkManager
...
/org/freedesktop/NetworkManager
/org/freedesktop/NetworkManager/ActiveConnection
...
/org/freedesktop/NetworkManager/IP6Config/0
/org/freedesktop/NetworkManager/Settings/0
```

(3)
Hier die Liste aller Methoden, Eigenschaften und Signale zu einer *ausgewählten Schnittstelle* aus (2):

```
$ qdbus --system org.freedesktop.NetworkManager /org/freedesktop/NetworkManager

method QString org.freedesktop.DBus.Introspectable.Introspect()
method QDBusVariant org.freedesktop.DBus.Properties.Get(QString interface, QString propname)
...
property read bool org.freedesktop.NetworkManager.NetworkingEnabled
property read QString org.freedesktop.NetworkManager.Version
…

signal void org.freedesktop.NetworkManager.DeviceAdded(QDBusObjectPath)
signal void org.freedesktop.NetworkManager.DeviceRemoved(QDBusObjectPath)
…

method uint org.freedesktop.NetworkManager.state()
```

Alternativ können Sie auch hier mit dem Programm dbus-send arbeiten:

```
$ dbus-send --system --print-reply --dest=org.freedesktop.NetworkManager /org/freedesktop/NetworkManager
org.freedesktop.DBus.Introspectable.Introspect
```

(4)
So gelingt das Auslesen ausgewählter Netzwerk-Eigenschaften zum Beispiel für den Netzwerk-Manager. Das setzt aber voraus, dass Sie die im Punkt (3) ermittelten Namen der Eigenschaften bereits genau kennen:

```
$ qdbus --system org.freedesktop.NetworkManager /org/freedesktop/NetworkManager org.freedesktop.NetworkMa-
nager.Version
0.9.8.8
```

```
$ qdbus --system org.freedesktop.NetworkManager /org/freedesktop/NetworkManager org.freedesktop.NetworkMa-
nager.NetworkingEnabled
true
```

Wesentlich komfortabler als in der Konsole erarbeiten Sie sich die Informationen zu den am D-Bus angemeldeten Anwendungen entweder mit dem Gambas-Programm *DBusExplorer* von Benoît Minisini und Fabien Bodard oder dem Konsolen-Programm d-feet.

#### 24.9.0.2.2 Programm DBus-Explorer

So arbeiten Sie erfolgreich mit dem Gambas-Programm DBusExplorer:

- Wählen Sie zuerst den gewünschten Bus aus (Session-Bus oder System-Bus).
- Dann liest ein *Doppel-Klick* auf den ausgewählten Dienst das XML-Dokument aus und zeigt die Schnittstellen-Beschreibung vollständig an, nachdem Sie alle Einträge aufgeklappt haben.
- Die Legende informiert Sie über die Bedeutung der einzelnen Farben.

- Mit dem grünen Pfeil-nach-unten-Button werden beispielsweise alle Einträge aufgeklappt angezeigt.
- Mit dem grünen Pfeil-nach-oben-Button dagegen werden alle Einträge – bis auf den ersten Bezeichner – zugeklappt.
- In der Statuszeile erkennen Sie bei einer ausgewählten Methode oder einem Signal den Datentyp der Argumente, die von der Methode oder dem Signal zurückgegeben werden.

Das Projekt von Fabien Bodard und Benoît Minisini wurde vom Autor so adaptiert, dass u.a. in der GUI nun auch Schnittstellen (Interface) farblich gekennzeichnet werden:

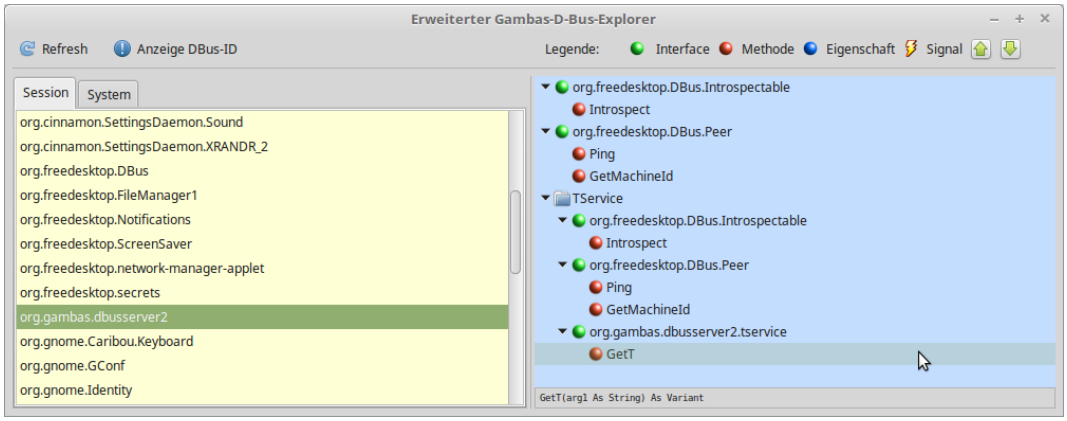


Abbildung 24.9.0.2.1: DBusExplorer

Das erweiterte Projekt DBus-Explorer finden Sie im Download-Bereich.

### 24.9.0.2.3 Programm d-feet

Ein Programm mit ähnlicher Funktionalität ist *d-feet*, das ebenso auf Schnittstellen-Spurensuche für den D-Bus geht. Das Programm ist der Favorit des Autors. Das Besondere an diesem Programm ist die Möglichkeit, dass nach einem Doppel-Klick auf eine Eigenschaft (Abbildung 24.9.0.2.2) deren aktueller Wert hinter der Eigenschaft angezeigt wird. Bei den Methoden dagegen öffnet sich ein Dialog, in dem geeignete Argumente für die Parameter eingegeben werden können, mit denen die ausgewählte Methode anschließend aufgerufen werden kann. Das sollten Sie aber nur dann tun, wenn Sie sehr genau wissen, was die Methode leistet!

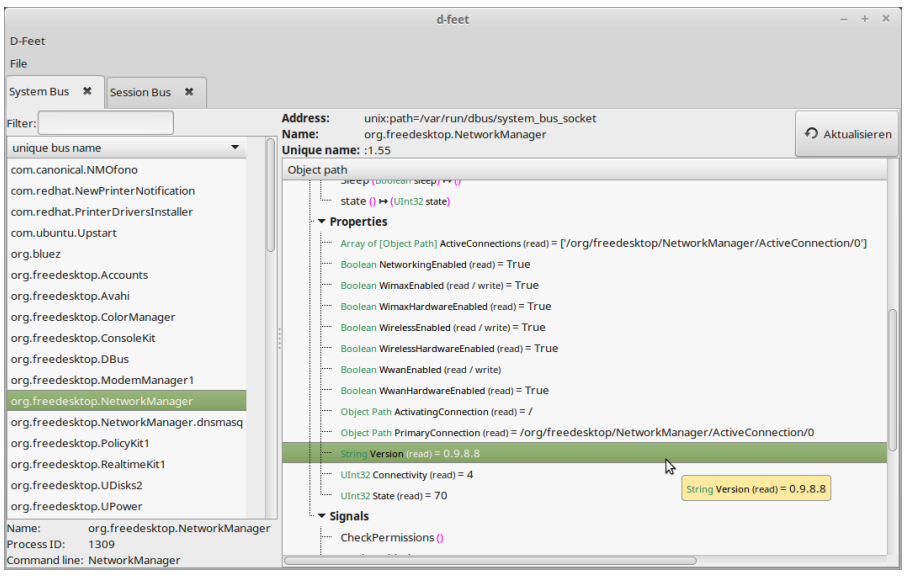


Abbildung 24.9.0.2.2: Programm d-feet

Vom Autor werden für eine Introspection vorrangig die beiden Programme d-feet und dbus-monitor verwendet. Für das Programm d-feet kann Folgendes empfohlen werden:

- Programm d-feet starten
- D-Bus passend auswählen – hier Session
- D-Bus-Anwendung auswählen – org.gtk.Private.UDisks2VolumeMonitor
- D-Bus-Objekt auswählen – /org/gtk/Private/RemoteVolumeMonitor
- Interfaces ausklappen (1. Eintrag: org.freedesktop.DBus.Introspectable)
- Methods ausklappen (Eintrag: Introspect() → String xml_data)
- Doppelklick auf die Methode Introspect()
- Das sich öffnende Fenster maximieren
- Die Inspektion mit 'Execute' ausführen und das XML-Dokument lesen

Mit den korrekten Adressen und einer Schnittstellen-Beschreibung für eine bestimmte d-bus-fähige Anwendung können Sie zum Beispiel deren Eigenschaften auslesen oder die Anwendung über Methoden-Aufrufe fern-steuern. Mit folgendem Befehl (Methode Stop; ohne Argument) fahren Sie zum Beispiel Ihren Computer über den D-Bus (!) herunter:

```
$ dbus-send --system --print-reply --dest=org.freedesktop.ConsoleKit /org/freedesktop/ConsoleKit/Manager \
  org.freedesktop.ConsoleKit.Manager.Stop
```

So erfahren Sie den Host-Namen für das System:

```
$ qdbus --system org.freedesktop.NetworkManager \
  /org/freedesktop/NetworkManager/Settings \
  org.freedesktop.NetworkManager.Settings.Hostname
linux
```

### 24.9.0.2.4 Programm DBusView

Eine Übersicht zu allen am D-Bus registrierten Programmen erzeugt auch das Gambas-Programm DBusView, das im Kapitel 24.9.2.1 vorgestellt wird:

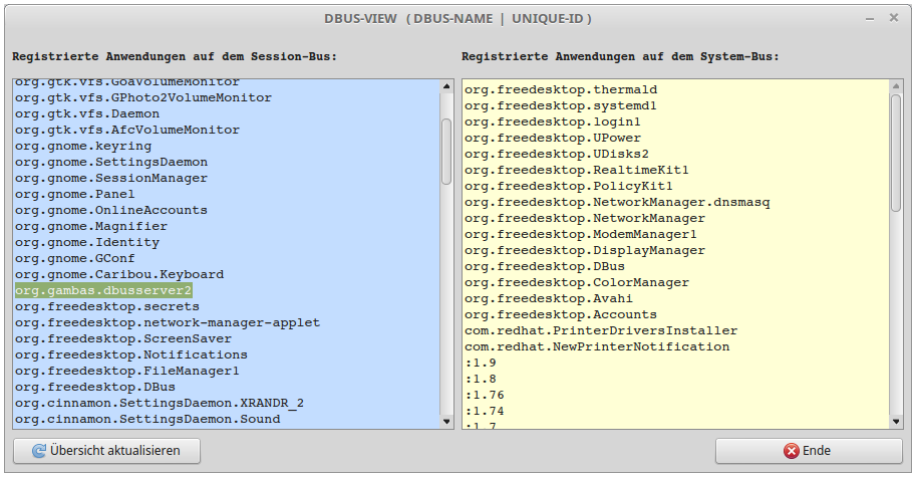


Abbildung 24.9.0.2.3: Registrierte Anwendungen auf dem Session-D-Bus und auf dem System-D-Bus

### 24.9.0.2.5 Monitoring D-Bus

Das Diagnose-Programm *dbus-monitor* gibt nach dem Start alle Nachrichten aus, die von den am Bus registrierten Anwendungen gesendet werden. Die sind jedoch so zahlreich, dass Sie geeignete Filter für die Ausgabe verwenden sollten, wenn Sie zum Beispiel nur der Nachrichten-Typ interessiert:

```
$ dbus-monitor
```

Beispiel mit einem einfachen Filter:

```
$ dbus-monitor | grep 'type'
   string "eavesdrop=true,type='method_call'"
   string "eavesdrop=true,type='method_return'"
   string "eavesdrop=true,type='error'"
...
```

So richtig interessant wird es dann, wenn Sie sich zum Beispiel die Signale ansehen, die von der Anwendung *org.gtk.vfs.UDisks2VolumeMonitor* mit dem Objekt-Pfad */org/gtk/Private/RemoteVolumeMonitor* gesendet werden, wenn ein USB-Stick eingesteckt wird. Diese Informationen erhalten Sie zum Beispiel für das Signal mit dem Signal-Namen "VolumeAdded" (der Signal-Name wird auch Member genannt), aus denen Sie auch die Signaturen der Argumente des Signals ablesen können:

```
$ dbus-monitor
…
signal time=1523464912.402058 sender=:1.28 -> destination=(null destination) serial=60 path=/org/gtk/Priva-
te/RemoteVolumeMonitor; interface=org.gtk.Private.RemoteVolumeMonitor; member=VolumeAdded
   string "org.gtk.vfs.UDisks2VolumeMonitor"
   string "0xe01330"
   struct {
      string "0xe01330"
      string "2_GB_P"
      string ". GThemedIcon drive-removable-media-usb drive-removable-media drive-removable drive"
      string ". GThemedIcon drive-removable-media-usb-symbolic … drive-removable drive"
      string ""
      string ""
      boolean true
      boolean true
      string "0xe23480"
      string ""
      array [
         dict entry(
            string "class"
            string "device"
         )
         dict entry(
            string "unix-device"
            string "/dev/sdd1"
         )
         dict entry(
            string "label"
            string "2_GB_P"
         )
         dict entry(
            string "uuid"
            string "38ea6e0b-a161-401d-a673-11c06cf1229e"
         )
      ]
      string "gvfs.time_detected_usec.1523464912399238"
      array [
      ]
   }
```

Grundlegende Informationen zum Thema D-Bus-Signatur können Sie im folgenden Kapitel nachlesen.

#### 24.9.0.2.6 Gambas-Programm: Introspection

Brauchen Sie in einem Gambas-Programm die Ergebnisse einer D-Bus-Introspection, dann benötigen Sie den D-Bus-Namen der zu inspizierenden Anwendung und den D-Bus-Objekt-Pfad. Mit diesem Quelltext (FIntrospection.class) gelingt die Intospection auf dem Session-D-Bus:

```
' Gambas class file

Public Sub Form_Open()
  FIntrospection.X = 150 ' possibly adapt
  FIntrospection.Y = 200
  FIntrospection.Caption = "Introspection D-Bus-Server - Object '/MSService'"
  txaIntrospection.ReadOnly = True
End

Public Sub Form_Show()

  Dim sDBusApplicationName, sDBusObjectPath, sXMLText As String

  sDBusApplicationName = "session://" & "org.gambas.dbusserver2"
  sDBusObjectPath = "/MSService"
  Try sXMLText = DBus[sDBusApplicationName]._Introspect(sDBusObjectPath)
  txaIntrospection.Insert(RTrim(sXMLText))
  txaIntrospection.Pos = txaIntrospection.Length
End

Public Sub Form_KeyPress()
  If Key.Code = Key.ESC And FIntrospection.Closed = False Then FIntrospection.Delete()
  Endif
End
```

Im Hauptprogramm könnten Sie das Fenster mit dem Ergebnis der D-Bus-Introspection anzeigen:

```
Public Sub btnIntrospection_Click()
  FIntrospection.Show()
End
```

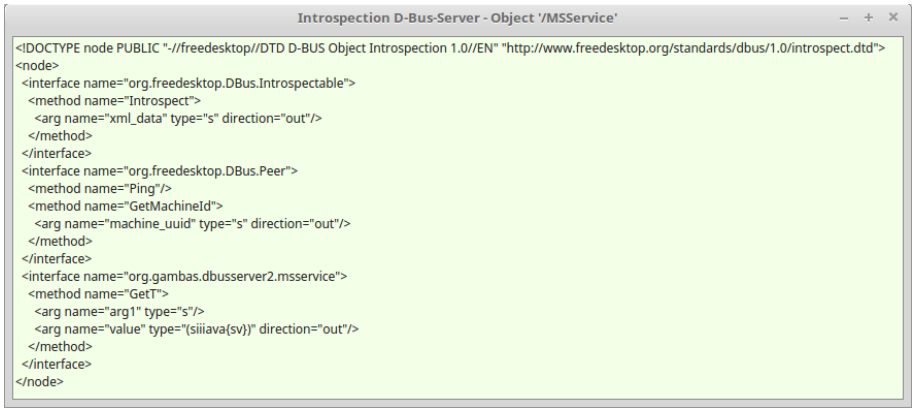


Abbildung 24.9.0.2.4: Ergebnis der Introspection