24.9.6.0 DBusSignal

Die Klasse *DBusSignal* (gb.dbus) ermöglicht es, jede Nachricht vom Typ *Signal* zu *empfangen*, das von einer am D-Bus-Bus registrierten Anwendung auf dem D-Bus gesendet wird. Die Klasse kann erzeugt werden. Die Klasse DBusSignal besitzt nur eine Eigenschaft und ein Ereignis.

24.9.6.0.1 Eigenschaft

Die boolsche Eigenschaft *Enabled* der Klasse DBusSignal ist True, wenn der Signal-Observer eingeschaltet ist. Der Standard-Wert ist True. Sie können den Wert auch setzen und auslesen.

24.9.6.0.2 Ereignis

Das Ereignis *Signal (Signal As String, Arguments As Variant[ ])* wird ausgelöst, wenn das DBus-Signal-Objekt ein D-Bus-Signal eines ausgewählten Senders empfängt. Für die beiden Parameter gilt:

- Signal ist der DBusSignal-Name, der auch Member genannt wird.
- Arguments ist ein Variant-Array mit allen Signal-Argumenten.

24.9.6.0.3 New DBusSignal(...)

So erzeugen Sie ein neues DBus-Signal-Objekt:

```
Dim hDBusSignal As DBusSignal

hDBusSignal = New DBusSignal ( Connection As DBusConnection, Interface As String [ , Every As Boolean ] )
As "SignalEventName"
```

- Connection bezieht sich auf den Session-Bus oder den System-Bus, zu dem verbunden wird.
- Interface ist die Schnittstelle, in der das Signal erzeugt und abgesendet wird.
- Setzen Sie das optionale Argument *Every* auf TRUE, wenn Sie die von allen Anwendungen auf dem D-Bus gesendeten Signale abfangen wollen. Sonst erhalten Sie nur die Signale, die speziell an Ihre Anwendung gesendet wurden.

24.9.6.0.4 D-Bus-Signal abfangen

Im folgenden Bespiel wird das D-Bus-Signal mit dem Signal-Namen 'StateChanged' abgefangen, das vom D-Bus-Ojekt "/org/freedesktop/NetworkManager" auf dem System-D-Bus gesendet wird, wenn sich der Status (on/off) des Netzwerks ändert.

```
Public hDBusSignalNM As DBusSignal
Private $iNetworkState As Integer
Private $cDBus As DBusConnection
Private $sInterface As String

Public Sub Form_Open()
…
  $cDBus = DBus.System
  $sInterface = "org.freedesktop.NetworkManager"
  hDBusSignalNM = New DBusSignal($cDBus, $sInterface, True) As "ObservedNMSignal"
…
End
```

Die Signatur des Signals ist "i" – es wird ein Integerwert gesendet. Wird das Signal mit dem oben angegebenen Signalnamen abgefangen, dann liest man dessen Wert aus dem Variant-Array der Argumente aus und speichert es zum Beispiel in einer Variablen:

```
Public Sub ObservedNMSignal_Signal(Signal As String, Arguments As Variant[])
  If Signal = "StateChanged" Then
     $iNetworkState = Arguments[0] ' The array has exactly one element!
  Endif
End
```

Der Signal-Observer wird stets ausgeschaltet, bevor das Programm-Fenster geschlossen wird:

```
Public Sub btnClose_Click()
  If hDBusSignalNM Then hDBusSignalNM.Enabled = False
  FMain.Close()
End
```

24.9.6.0.5 D-Bus-Signal senden

In der Revision #7146 wurde der Komponente gb.dbus die Fähigkeit hinzugefügt, auch D-Bus-Signale zu senden.

So funktioniert es:

- Alle in einem D-Bus-Objekt definierten Ereignisse werden zu D-Bus-Signalen.
- Der Name des Ereignisses muss der Name des Interfaces sein, bei dem alle Punkte durch Unterstriche "_" ersetzt werden – gefolgt von einem Unterstrich und dem Namen des Signals.
- Wenn es sich bei dem Ereignis um die Standard-Schnittstelle handelt, dann ist der Signal-Name ausreichend.
- Vergessen Sie nicht, beim notwendigen Aufruf der Register()-Methode den Namen der Schnittstelle hinzuzufügen!

Um ein D-Bus-Signal auf einem bestimmten D-Bus zu senden, verwenden Sie eine der beiden Methoden: DBus.Session.Raise(...) oder DBus.System.Raise(...).

Die DBus-Raise(...)-Methode benötigt 3 Argumente:

- 1. Argument: DBus-Objekt
- 2. Argument: Signalname. Das ist der Name des Ereignisses, bei dem alle Unterstriche "_" durch einen Punkt "." ersetzt werden.
- 3. Argument: Ein *optionales* Array von Signal-Argumenten. Wenn Sie nur das Signal über den Signal-Namen abfangen wollen und Ihnen das als Information ausreicht, dann können Sie auf die Auswertung des Array der Signal-Argumente verzichten.

Eine Signal-Signatur müssen Sie nicht festlegen, weil diese nicht direkt verwendet wird. Es ist aber sinnvoll, es trotzdem zu tun. So stellen Sie für sich sicher, dass die (Pseudo-)Signal-Signatur und die Signal-Argumente übereinstimmen! Gut zu wissen: Die Daten-Typen der einzelnen Signal-Argumente im o.a. Array werden automatisch in D-Bus-Datentypen konvertiert.

Bevor Sie ein D-Bus-Signal senden können, müssen Sie es in einer eigenen Klassendatei definieren.

Signal-Definition

Die Aufbereitung eines in der Anwendung MyProject zu sendenden Signals erfolgt zum Beispiel in der Klassen-Datei MySignal.class. Den Namen der Klassen-Datei und den Signalnamen – im folgenden Beispiel State – können Sie selbst festlegen:

```
' Gambas class file (MySignal.class)

Inherits DBusObject
Create Static

'' Signatur: "s"
'' This signal has exactly 1 argument 'Flag' of the data type string
Event org_gambas_MyProject_MySignal_State(Flag As String)

End
```

Wie Sie erkennen, hat das Signal 'State' genau ein Argument mit dem Daten-Typ und folgt damit der Signatur "s".

Signal senden

Der Quelltext für die Datei *FMain.class* des Projektes *MyProject* wird vollständig angegeben. Die wichtigsten Passagen werden farbig hervorgehoben:

```
' Gambas class file

Private hDBusObject As MySignal
Private hDBusSignal As DBusSignal

Public Sub Form_Open()
```

```
  hDBusObject = New MySignal

  FMain.Resizable = False
  FMain.Caption = ("Waiting for a D-Bus signal ...")
  DBus.Unique = True

  DBus.Session.Register(hDBusObject, "/MySignal")
  hDBusSignal = New DBusSignal(DBus.Session, Null, True)

  SendSignal("on")

End

Private Sub SendSignal(sState As String)

  Dim sSignalName As String
  Dim aArguments As New Variant[]

  sSignalName = "org.gambas.MyProject.MySignal.State"
  aArguments = [sState]

  DBus.Raise(hDBusObject, sSignalName, aArguments)

End


Public Sub Form_Close()

  SendSignal("off")
    If hDBusSignal Then hDBusSignal.Enabled = False
    If DBus.IsRegistered(hDBusObject) Then DBus.Session.Unregister(hDBusObject)
  FMain.Close()

End
```

Damit Sie das Projekt erfolgreich erproben können, wird auch der Quelltext für den Signal-Empfänger präsentiert:

```
' Gambas class file

Private $hDBusSignal As DBusSignal

Public Sub Form_Open()

  FMain.Resizable = False
  FMain.Caption = ("Waiting for a D-Bus signal ...")
  $hDBusSignal = New DBusSignal(DBus.Session, Null, True) As "ObservedSignal"
  SetLEDColor(picStatus, "red")
  txaReport.Insert(gb.NewLine)

End

Public Sub ObservedSignal_Signal(Signal As String, Arguments As Variant[])
  If Lower(Signal) = "state" And If Arguments[0] = "on" Then
     txaReport.Insert("    " & ("The data server is online!") & gb.NewLine)
     SetLEDColor(picStatus, "green")
     SendSound("on.wav")
  Else If
     Lower(Signal) = "state" And If Arguments[0] = "off" Then
     txaReport.Insert("    " & ("The data server is down!") & gb.NewLine)
     SetLEDColor(picStatus, "red")
     SendSound("off.wav")
  Endif
End

'' Sends a sound<br>
'' Sound: Name of the sound file in the project folder 'sounds'.<br>
'' Play back audio files on a PulseAudio sound server - the player is 'paplay'
Public Sub SendSound(SoundFileName As String)
  If System.Exist("paplay") Then
     Shell "paplay " & Application.Path &/ "sounds" &/ SoundFileName
  Endif
End

Private Sub SetLEDColor(picBox As PictureBox, sLEDColor As String)
  picBox.Picture = Picture["LED/led_" & sLEDColor & ".svg"]
End

Public Sub Form_Close()
  If $hDBusSignal Then $hDBusSignal.Enabled = False
  FMain.Close()
End
```

Der Server sendet beim Starten und beim Schließen des Programm-Fensters stets das D-Bus-Signal 'State', dass seinen Status über das Signal-Argument mit den Werten 'on' oder 'off' signalisiert. Der Client, der auf das Signal wartet, kann in geeigneter Weise auf den übermittelten Status reagieren.

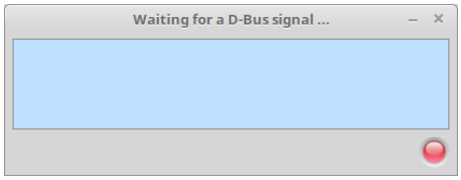

Abbildung 24.9.6.0.1: Der Server ist offline

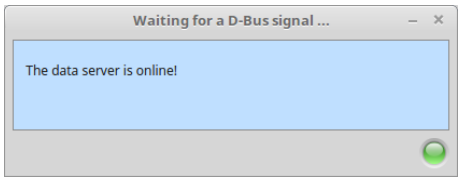

Abbildung 24.9.6.0.2: Der Server ist online

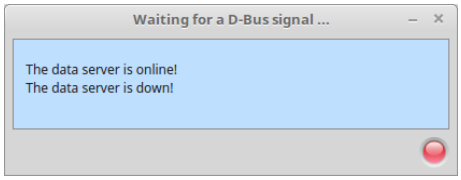

Abbildung 24.9.6.0.3: Der Server ist offline

Hinweis:

Wenn Sie mehr Informationen darüber brauchen, welche Meta-Daten das abgefangene Signal neben dem eigentlichen Inhalt besitzt, verwenden Sie mit Erfolg die Klasse *DBusObserver*.

In den Projekten zur Klasse DBusSignal in den folgenden drei Kapiteln erfahren Sie, wie man Signale abfängt, deren Inhalt auswertet und wie man Signale sendet.