24.9.2 DBus

Die Klasse DBus (gb.dbus) verwaltet die Verbindung einer Anwendung zum Session-Bus oder zum System-Bus und erbt von der virtuellen Klasse _DBus.

24.9.2.1 Eigenschaften

Die Klasse *DBus* verfügt über die folgenden sechs Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Debug	Boolean	Ist True, wenn Debugging-Meldungen nach DBus.Debug = True ausgegeben werden oder setzt den boolschen Wert.
Name	String	Der Name der D-Bus-Anwendung wird gesetzt oder zurückgegeben. Standardmäßig hat der D-Bus-Name folgende Syntax: org.gambas.<xxx>, wobei <xxx> der Wert der Eigenschaft *Application.Name* ist. Beachten Sie: Der D-Bus-Name wird *automatisch* normalisiert und an die D-Bus-Spezifikation angepasst.
Null	Objekt	Es wird ein besonderes Objekt zurückgegeben, dass den D-Bus-Wert NULL repräsentiert.
Session	DBusConnection	Kennzeichnet die Verbindung zwischen einer Anwendung und dem Session-Bus vom Typ DBusConnection.
System	DBusConnection	Kennzeichnet die Verbindung zwischen einer Anwendung und dem System-Bus vom Typ DBusConnection.
Unique	Boolean	Ist True, wenn eine Anwendung am D-Bus mit einer einzigartigen ID registriert ist oder setzt die Unique-Eigenschaft. Beachten Sie bitte die folgenden Hinweise.

Tabelle 24.9.2.1.1 : Eigenschaften der Klasse DBus

Hinweise zur Eigenschaft DBus.Unique:

Für den Fall, dass Sie die Eigenschaft *DBus.Unique* auf True setzen, ist der Name der Anwendung

```
sDBusApplicationName  = org.gambas.projektname
```

Für den alternativen Fall *DBus.Unique* = *False* (das ist der Standard-Wert) wird daraus aber

```
sDBusApplicationName  = org.gambas.projektname-PID  ' PID = Process Identification Number
```

Da nicht notwendigerweise bekannt ist, welchen aktuellen Wert *DBus.Unique* zum Beispiel für einen zu nutzenden D-Bus-Server besitzt, können Sie im Quelltext für einen D-Bus-Client die folgende Funktion verwenden, um den korrekten Namen des Servers auf dem D-Bus zu ermitteln:

```
Private Function GetDBusApplicationName(GBAppName As String) As String

  Dim aList As String[]
  Dim sElement  As String

  aList = DBus.Session.Applications.Sort(gb.Natural)

  For Each sElement In aList
    If sElement Begins "org.gambas." & GBApp.Name Then

       Return sElement

    Endif
  Next

End
```

24.9.2.2 Methoden

Die Klasse *DBus* besitzt vier Methoden, die in der folgenden Tabelle angegeben und beschrieben werden:

Methode	Beschreibung
IsRegistered (Object As DBusObject)	Die boolsche Funktion gibt True zurück, wenn das als Parameter übergebene Objekt am D-Bus registriert ist.
Register (Object As DBusObject, Path As String [, Interfaces As String[]])	Exportiert ein DBus-Objekt zum D-Bus. Für die Parameter gilt: Object ist das zu exportierende D-Bus-Objekt, Path ist der D-Bus-Pfad für das D-Bus-Objekt und Interface (optional) ermöglicht die Angabe der Namen von zusätzlichen Schnittstellen, die für das verwendete D-Bus-Objekt implementiert wurden. Details können Sie in der Dokumentation unter *DBusConnection.Register()* nachlesen.
Unregister (Object As DBusObject)	Meldet das im Parameter übergebene DBus-Objekt vom D-Bus ab und ist das Äquivalent zu *DBus.Session.Unregister()*.
Raise (Object As DBusObject, Signal As String [, Arguments As Variant[]])	Löst ein D-Bus-Signal aus. Für die Parameter gilt: Object ist das Objekt, welches das Signal sendet, Signal ist der Name des gesendeten Signals und Arguments repräsentiert die (optionalen) Argumente des Signals in einem Variant-Array.

Tabelle 24.9.2.2.1 : Methoden der Klasse DBus

Hinweise:

- Ein Gambas-Programm wird *automatisch* mit dem Session-D-Bus verbunden, wenn die Komponente *gb.dbus* geladen wird! Das hat Nichts mit der oben beschriebenen Registrierung zu tun. Die Registrierung müssen Sie nur dann vornehmen, wenn Sie eigene Objekte zum D-Bus exportieren wollen, um die in den D-Bus-Objekten programmierten Methoden und Eigenschaften anderen d-bus-fähigen Anwendungen als Dienst zur Verfügung zu stellen. Wenn Sie in Ihrer Gambas-Anwendung nur die auf dem D-Bus angebotenen Dienste nutzen wollen, dann müssen Sie Ihre Gambas-Anwendung nicht am D-Bus registrieren.
- Die beiden Methoden Register(...) und Unregister(...) beziehen sich nur auf D-Bus-Objekte in Gambas-Programmen!
- Um festzustellen, ob eine bestimmte Anwendung aktuell auf dem D-Bus registriert ist, können Sie den folgenden Quelltext einsetzen, wenn Sie den Anwendungsnamen anpassen:

```
If DBus.Session.Applications.Exist($sDBusName) Then
```

- Wenn es erforderlich ist, können Sie die durch Ihr Gambas-Programm exportierten Objekte wieder vom D-Bus abmelden:

```
Public Sub Form_Close()

  If DBus.IsRegistered($hDBusObject) Then DBus.Session.Unregister($hDBusObject)
  FMain.Close()

End
```

Die Antwort auf die Frage 'Was wird zum D-Bus exportiert?' hängt davon ab, ob Sie in der Register-Methode das optionale Argument *Interfaces As String[]* verwenden oder nicht.

Registrierung ohne Schnittstellen (Interfaces):

- Öffentliche Methoden, deren Name nicht mit einem Unterstrich beginnt. Deren optionale Argumente und der Rückgabetyp der Methode können in einen D-Bus-Datentyp umgewandelt werden.
- Öffentliche Eigenschaften, deren Name im Inneren keinen Unterstrich aufweist und deren Typ in einen D-Bus-Datentyp umgewandelt werden kann.

Sobald Sie mindestens ein Objekt exportieren, wird Ihre Anwendung auf dem D-Bus unter dem Namen org.gambas.<Anwendungsname> registriert. Für <Anwendungsname> können Sie stets die Eigenschaft *Application.Name* einsetzen.

Alle Methoden und Eigenschaften, deren Name mit einem im Interface-Argument spezifizierten Schnittstellennamen beginnt – wobei die Punkte durch Unterstrich ersetzt werden und ein zusätzlicher Unterstrich folgt – werden unter dieser Schnittstelle exportiert.

Registrierung mit mindestens einer Schnittstelle (Interface):

- Durch eine Schnittstelle mit dem Namen *org.gambas.<Anwendungsname>.<Klassen-Name>* werden nur deren Methoden und Eigenschaften der angegebenen Klasse exportiert.
- Sie können auch mehrere Schnittstellen angeben, wenn Sie in der Register(...)-Methode das optionale Argument *Interfaces As String[]* verwenden! Das Array enthält die Namen der Schnittstellen.

In den Kapiteln 24.9.8.1 und 24.9.8.2 finden Sie zwei (Server-)Projekte, die Objekte zum D-Bus exportieren.

24.9.2.3 Konstanten

Die folgenden vier Konstanten der Klasse *DBus* repräsentieren jeweils einen Nachrichten-Typ, der einen der folgenden Integer-Werte annehmen kann:

- DBus.Method (1),
- DBus.Reply (2),
- DBus.Error (3),
- DBus.Signal (4).

24.9.2.4 Beispiel – DBus[ ]

Es wird mit DBus[…] ein Objekt zurückgegeben, das eine zum D-Bus verbundene Anwendung repräsentiert, denn die Klasse DBus ist statisch und verhält sich wie ein (read-only-)Array:

```
Dim hDBusApplication As DBusApplication
hDBusApplication = DBus [ Application As String ]
```

- Wenn das Argument *Application* mit "system://" beginnt, dann wird die Anwendung auf dem System-Bus gesucht.
- Wenn das Argument *Application* mit "session://" beginnt, dann wird die Anwendung auf dem Session-Bus gesucht.

Der Standard-Wert für den Präfix des Arguments *Application* ist "session://".

Beispiele:

```
Dim hDBusApplication As DBusApplication
Dim hConnection As DBusConnection
hConnection = DBus.System
hDBusApplication = DBus["system://org.freedesktop.ConsoleKit"]
```

```
Dim hDBusApplication As DBusApplication
hDBusApplication = DBus["session://org.Cinnamon.LookingGlass"] oder
hDBusApplication = DBus["org.Cinnamon.LookingGlass"] Standard-Parameter 'session:\\' intern gesetzt
```