24.9.8.2 Projekte – D-Bus-Objekt erzeugen, exportieren und verwenden

In diesem Kapitel wird Ihnen u.a. ein Gambas-Programm vorgestellt, das einen D-Bus-Server realisiert, der einen Service für d-bus-fähige Programme anbietet, den ein zum Session-D-Bus exportiertes D-Bus-Objekt bereitstellt. Der implementierte Service kann so beschrieben werden:

- Wenn die implementierte Methode *GetT(ShortScale As String)* mit einem Parameter vom Daten-Typ String von einem d-bus-fähigen Client aufgerufen wird, dann wird ein Rückgabewert mit einem komplexen Daten-Typ zurückgegeben.
- Der komplexe Datentyp – beschrieben in der Signatur "(siiiava{sv})" – wurde so gewählt, dass ein breites Anwendungsspektrum für Ihre eigenen Projekte abgedeckt wird.

Der komplexe Datentyp für den Rückgabewert der Methode *GetT(ShortScale As String)* hat die Signatur "(siiiava{sv})". Diese Signatur wird verständlich, wenn Sie sich die (Teil-)Daten des Rückgabewertes mit ihren Datentypen ansehen:

```
String          s0     ' Sensor Number/Type: 70358 KNX T-B-UP
Integer 1       i1     ' Day      15
Integer 2       i2     ' Month    12
Integer 3       i3     ' Year     2017

Variant-Array   av
Array[0]        i4     ' Hour     15
Array[1]        i5     ' Minute   24
Array[2]        i6     ' Second   37
Array[3]        s1     ' Time zone "LocalTime"

Collection      a{sv}  ' Key.Type=String, Value-Type=Variant
c["T"]          f1     ' Temperature -1.4
c["L"]          s2     ' Label       °
c["S"]          s3     ' Scale       C
```

Da der Funktionswert für *GetT(ShortScale As String)* kein nativer Datentyp ist, müssen Sie sich um die Konvertierung der einzelnen Gambas-Datentypen in D-Bus-Datentypen kümmern! Nutzen Sie die Tabelle unter http://gambaswiki.org/wiki/doc/dbus#t10 um zu erfahren, wie Gambas-Datentypen in D-Bus-Datentypen konvertiert werden und umgekehrt.

Signaturen wurden bereits in den Kapiteln 24.9.0.3 und 24.9.6.3 beschrieben.

24.9.8.2.1 Projekt Server

Der Quelltext für den D-Bus-Server besteht bei einem komplexen Datentyp für den Rückgabewert der implementierten Methode aus mindestens 3 Klassen-Dateien, deren Namen Sie frei festlegen können:

(A) In der Datei RValue.class wird der komplexe Datentyp für den Rückgabewert der Methode GetT(…) über seine Signatur-Konstante definiert.
(B) Der Service wird in einem D-Bus-Objekt implementiert, das in einer speziellen Klassen-Datei MSService.class beschrieben wird. Die Klasse verwendet den Rückgabewert der Methode GetT(…) aus (A).
(C) In der Startklasse FMain.class wird das D-Bus-Objekt "/MSService" aus (B) zum Session-D-Bus exportiert, damit d-bus-fähige Programme den Service nutzen können.

Zuerst erzeugen Sie eine neue Klasse RValue.class, die exportiert werden soll (Export) und von der Klasse DBusVariant (Inherits DBusVariant). Dann definieren Sie die benötigte Signatur als Konstante. Der relevante Quelltext der Klasse RValue.class in den ersten 6 Zeilen ist sehr kurz. Alle weiteren Zeilen sind eine Beschreibung der Signatur.

```
' Gambas class file

Export
Inherits DBusVariant

Public Const Signature As String = "(siiiava{sv})"

' Signature = "(siiiava{sv})"
'-------------------------------------------------------------
' String          s0     ' Sensor Number/Type: 70358 KNX T-B-UP
' Integer 1       i1     ' Day      15
' Integer 2       i2     ' Month    12
' Integer 3       i3     ' Year     2017
```

```
' Variant-Array  av
' Array[0]       i4    ' Hour    15
' Array[1]       i5    ' Minute  24
' Array[2]       i6    ' Second  37
' Array[3]       s1    ' Time zone "LocalTime"

' Collection     a{sv} ' Key.Type=String, Value-Type=Variant
' c["T"]         f1    ' Temperature -1.4
' c["L"]         s2    ' Label       °
' c["S"]         s3    ' Scale       C

' The introspection shows the signature of the method and the type of the one argument

' 1. Case with the command 'dbus-send' in one console:
' dbus-send --session --print-reply --dest=org.gambas.dbusserver2 \
' /MSService org.freedesktop.DBus.Introspectable.Introspect \
' > tservice.introspection.xml

' 2. Case with code for introspection in DBusClient2:
' sXMLDocument = DBus[("session://" & "org.gambas.dbusserver2")]._Introspect("/MSService")
' Extract from the XML document:
'  ...
'   <interface name="org.gambas.dbusserver2.msservice">
'     <method name="GetT">
'       <arg name="arg1" type="s"/>
'       <arg name="value" type="(siiiava{sv})" direction="out"/>
'     </method>
'   </interface>
'  ...
```

Danach deklarieren Sie in der Datei MSService.class eine Methode GetT(...), deren Funktionswert vom Typ RValue ist. Gehen Sie dabei nach folgendem Konzept vor:

1. Zuerst erzeugen Sie in der Funktion *GetT(ShortScale As String) As RValue* ein neues Objekt hItem vom Daten-Typ RValue (Zeilen 14 und 59).
2. Dann weisen Sie allen (Teil-)Daten, die im Rückgabewert existieren, die vorgesehenen Werte zu (Zeilen 21 bis 30 sowie Zeile 35).
3. Danach weisen Sie der Eigenschaft hItem.Value das Array [s0, i1, i2, i3, aTime, cTemperature] zu, was der o.a. Signatur "(siiiava{sv})" folgt. Füllen Sie zuvor das Variant-Array (Zeile 31) mit den entsprechenden Werten und die Collection mit den geplanten Wert-Schlüssel-Paaren (Zeilen 55 bis 57).
4. Abschließend geben Sie das Objekt hItem mit 'Return hItem' als Funktionswert von GetT(…) zurück (Zeile 63).

Der folgende Quelltext in der Datei MSService.class setzt das Konzept um:

```
[1]  ' Gambas class file
[2]
[3]  Inherits DBusObject
[4]  Create Static
[5]
[6]  '' Definition of a signal:
[7]  '' This signal has exactly 1 argument 'sFlag' of the data type string
[8]  Event org_gambas_dbusserver2_MSService_OnOff(sFlag As String)
[9]
[10] '' Definition of a method:
[11] '' The method has exactly one argument.
[12] Public Function GetT(ShortScale As String) As RValue
[13]
[14]   Dim hItem As RValue
[15]   Dim s0, s1, s2, s3 As String
[16]   Dim i1, i2, i3, i4, i5, i6 As Integer
[17]   Dim f1 As Float
[18]   Dim aTime As Variant[]
[19]   Dim cTemperature As Collection
[20]
[21]   s0 = "Sensor 34 : 70358 KNX T-B-UP"
[22]
[23]   i1 = Day(Now())
[24]   i2 = Month(Now())
[25]   i3 = Year(Now())
[26]
[27]   i4 = Hour(Now())
[28]   i5 = Minute(Now())
[29]   i6 = Second(Now())
[30]   s1 = "LocalTime"
[31]   aTime = [i4, i5, i6, s1]
```

```
[32]
[33] ' The temperature value is read out in real operation from an RS232-AD converter
[34] ' A random value for the current temperature is generated here
[35]   f1 = Round(Rnd(-3, 2), -4)
[36]
[37]   Select Case ShortScale
[38]     Case "C"
[39]       s2 = "°"
[40]       s3 = "C"
[41]     Case "K"
[42]       f1 = f1 + 273.15
[43]       s2 = ""
[44]       s3 = "K"
[45]     Case "F"
[46]       f1 = (9 / 5) * f1 + 32
[47]       s2 = "°"
[48]       s3 = "F"
[49]     Default
[50]       s2 = "°"
[51]       s3 = "C"
[52]   End Select
[53]
[54]   cTemperature = New Collection
[55]   cTemperature.Add(f1, "Temperature") ' Temperature
[56]   cTemperature.Add(s2, "Label")       ' Label
[57]   cTemperature.Add(s3, "Scale")       ' Temperature scale
[58]
[59]   hItem = New RValue
[60] ' hItem.Value with complex data type:
[61]   hItem.Value = [s0, i1, i2, i3, aTime, cTemperature]
[62]
[63]   Return hItem
[64]
[65] End
```

Hinweis:

In Zeile 8 wird ein Signal mit einem Argument deklariert. Dieses Signal wird gesendet, wenn der Server abgeschaltet wird oder neu gestartet wird. Der Client fängt dieses Signal ab und reagiert entsprechend des ausgelesenen Wertes ("on" oder "off") des Arguments.

Im Quelltext von FMain.class sind die Registrierung des D-Bus-Objektes in der Zeile 15 sowie das Senden eines D-Bus-Signals in den Zeilen 22 und 41 die zentralen Anweisungen, nachdem in der Zeile 8 ein neues D-Bus-Objekt vom Typ *MSService* erzeugt wurde:

```
[1]  ' Gambas class file
[2]
[3]  Private hDBusObject As MSService
[4]  Private hDBusSignal As DBusSignal
[5]
[6]  Public Sub Form_Open()
[7]
[8]    hDBusObject = New MSService
[9]
[10]   FMain.Resizable = False
[11]   FMain.Caption = ("The data server is activated")
[12]   DBus.Unique = True
[13] ' For tests only: DBus.Debug = True
[14]
[15]   Try DBus.Session.Register(hDBusObject, "/MSService")
[16]   If Error Then
[17]      Message.Error("An instance of " & Application.Name & " already exists.")
[18]      FMain.Close()
[19]   Endif
[20]
[21]   hDBusSignal = New DBusSignal(DBus.Session, Null, True)
[22]   SendSignal("on")
[23]
[24] End
[25]
[26] Private Sub SendSignal(OnOff As String)
[27]
[28]   Dim sSignalName As String
[29]   Dim aArguments As New Variant[]
[30]
[31]   sSignalName = "org.gambas.dbusserver2.MSService.OnOff"
[32]   aArguments = [OnOff]
[33]
[34]   DBus.Raise(hDBusObject, sSignalName, aArguments)
[35]
```

```
[36] End
[37]
[38] Public Sub Form_Close()
[39]
[40]   SendSignal("off")
[41]
[42]   If hDBusSignal Then hDBusSignal.Enabled = False
[43]   If DBus.IsRegistered(hDBusObject) Then DBus.Session.Unregister(hDBusObject)
[44]
[45]   FMain.Close()
[46]
[47] End
```

24.9.8.2.2 Projekt Client

Der Quelltext für den Client dbusclient2 wird vollständig angegeben und anschließend kommentiert:

```
[1]  ' Gambas class file
[2]
[3]  Private $hDBusProxy As DBusProxy
[4]  Private $hDBusSignal As DBusSignal
[5]
[6]  Private $sDBusName As String
[7]  Private $sDBusObjectPath As String
[8]  Private $sReportLine As String
[9]  Private $hDate As Date
[10] Private $bInspectable As Boolean
[11] Private k As Integer
[12] Private $iFirst As Integer
[13]
[14] Public Sub Form_Open()
[15]
[16]   Dim sMessage As String
[17]
[18]   DBus.Debug = True
[19]   FMain.Resizable = False
[20]   FMain.Caption = ("Remote data enquiry via D-Bus")
[21]   Application.MainWindow = FMain
[22]
[23]   cmbScale.Add("Celsius")
[24]   cmbScale.Add("Kelvin")
[25]   cmbScale.Add("Fahrenheit")
[26]   cmbScale.Index = 0
[27]
[28]   $sDBusName = "org.gambas.dbusserver2"
[29]   $sDBusObjectPath = "/MSService"
[30]
[31] If Not DBus.Session.Applications.Exist($sDBusName) Then
[32]      sMessage = ("There is no suitable data server on the session bus!")
[33]      sMessage &= "<center><font color='red'>"
[34]      sMessage &= ("The program is terminated.")
[35]      sMessage &= "</font></center>"
[36]      Message.Warning(sMessage)
[37]      FMain.Close()
[38]   Else
[39]     $hDBusProxy = DBus[$sDBusName][$sDBusObjectPath]
[40]     SetLEDColor(picStatus, "green")
[41]     $bInspectable = True
[42]     $hDBusSignal = New DBusSignal(DBus.Session, Null, True) As "ObservedSignal"
[43]   Endif
[44]
[45] End
[46]
[47] Public Sub ObservedSignal_Signal(Signal As String, Arguments As Variant[])
[48]
[49]   If Lower(Signal) = "onoff" And If Arguments[0] = "on" Then
[50]      If $iFirst = 0 Then Inc $iFirst
[51]      txaReport.Insert(gb.NewLine)
[52]      txaReport.Insert("   " & ("The data server is online!") & gb.NewLine)
[53]      txaReport.Pos = txaReport.Length
[54]      $bInspectable = True
[55]      SetLEDColor(picStatus, "green")
[56]      SendSound("on.wav")
[57]   Endif
[58]   If Lower(Signal) = "onoff" And If Arguments[0] = "off" Then
[59]      txaReport.Insert(gb.NewLine & gb.NewLine)
[60]      txaReport.Insert("   " & ("The data server is down!") & gb.NewLine)
[61]      txaReport.Pos = txaReport.Length
[62]      $bInspectable = False
[63]      SetLEDColor(picStatus, "red")
[64]      SendSound("off.wav")
[65]   Endif
[66]
```

```
[67] End
[68]
[69] Private Sub GetData()
[70]
[71] '  <interface name="org.gambas.dbusserver2.msservice">
[72] '    <method name="GetT">
[73] '      <arg name="arg1" type="s"/>
[74] '      <arg name="value" type="(iiiava{sv})" direction="out"/>
[75] '    </method>
[76] '  </interface>
[77]
[78]   Dim R As Variant[] ' Result
[79]
[80]   R = $hDBusProxy.GetT(Left(cmbScale.Text))
[81]
[82]   ' Print R[0]        ' Sensor Number/Type
[83]   ' Print R[1]        ' Day     15
[84]   ' Print R[2]        ' Month   12
[85]   ' Print R[3]        ' Year    2017
[86]   ' Print R[4][0]     ' Hour    15
[87]   ' Print R[4][1]     ' Minute  24
[88]   ' Print R[4][2]     ' Second   37
[89]   ' Print R[4][3]     ' Time zone "LocelTime"
[90]   ' Print R[5]["T"]   ' Temperature 22.4
[91]   ' Print R[5]["L"]   ' Label       °
[92]   ' Print R[5]["S"]   ' Scale       C
[93]
[94]   $hDate = Date(R[3], R[2], R[1], R[4][0], R[4][1], R[4][2])
[95]
[96]   txaReport.Insert(gb.NewLine)
[97]   If k = 0 Then
[98]      $sReportLine = "   " & R[0] & "   | " & R[4][3] & gb.NewLine & gb.NewLine
[99]      txaReport.Insert($sReportLine)
[100]       Inc k
[101]    Endif
[102]    $sReportLine = "   " & Format($hDate, "dd. mmmm yyyy  -  hh:nn:ss")
[103]    $sReportLine &= " Uhr"
[104]    $sReportLine &= "   |   " & ("Temperature") & " =  "
[105]    If cmbScale.Text = "Kelvin" Then
[106]       $sReportLine &= Format(R[5]["Temperature"], "0.#0") & " " & R[5]["Label"] & R[5]["Scale"]
[107]    Else
[108]       $sReportLine &= Format(R[5]["Temperature"], "+0.#0") & " " & R[5]["Label"] & R[5]["Scale"]
[109]    Endif
[110]    txaReport.Insert($sReportLine)
[111]    txaReport.Pos = txaReport.Length
[112]
[113]  End
[114]
[115]  Public Sub btnGetData_Click()
[116]    If DBus.Session.Applications.Exist($sDbusName) Then
[117]       SetLEDColor(picStatus, "green")
[118]       GetData()
[119]       $bInspectable = True
[120]    Else
[121]       SetLEDColor(picStatus, "red")
[122]    Endif
[123]  End
[124]
[125]  '' Sends a sound<br>
[126]  '' Sound: Name of the sound file in the project folder 'sounds'.<br>
[127]  '' Play back audio files on a PulseAudio sound server - the player is 'paplay'
[128]  Public Sub SendSound(SoundFileName As String)
[129]    If System.Exist("paplay") Then
[130]       Shell "paplay " & Application.Path &/ "sounds" &/ SoundFileName
[131]    Endif
[132]  End
[133]
[134]  Private Sub SetLEDColor(picBox As PictureBox, sLEDColor As String)
[135]    picBox.Picture = Picture["LED/led_" & sLEDColor & ".svg"]
[136]  End
[137]
[138]  Public Sub btnIntrospektion_Click()
[139]    If $bInspectable = True Then FIntrospection.Show()
[140]  End
[141]
[142]  Public Sub Form_Close()
[143]    If $hDBusSignal Then $hDBusSignal.Enabled = False
[144]    FMain.Close()
[145]  End
```

Kommentar:

- In Zeile 17 wird geprüft, ob es die Anwendung mit dem D-Bus-Namen org.gambas.dbusserver2

auf dem Session-Bus gibt. Ist das der Fall, wird nach der Zuweisung in der Zeile 31 in der Zeile 39 ein Proxy erzeugt und mit diesem gearbeitet.

- In der Prozedur GetData() in der Zeile 80 wird die im D-Bus-Objekt "/MSService" implementierten Methode GetT(ShortScale As String) mit einem der drei möglichen Argumente aufgerufen und der Rückgabewert mit dem komplexen Datentyp in die Variable R eingelesen.
- Anschließend werden alle (Teil-)Daten entsprechend der bekannten Signatur "(siiiava{sv})" in den Zeilen 94 bis 111 extrahiert und angezeigt.

24.9.8.2.3 Einsatz von Server und Client

Server

Abbildung 24.9.8.2.1: Server-GUI

Der Server wird gestartet, exportiert ein D-Bus-Objekt zum Session-D-Bus mit dem implementierten Service und wartet auf Anfragen von d-bus-fähigen Clients.

Der vorgestellte Client nutzt den angebotenen Service des Servers:

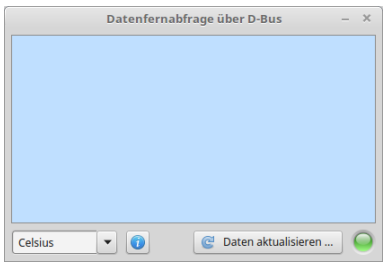

Abbildung 24.9.8.2.2: Client-GUI

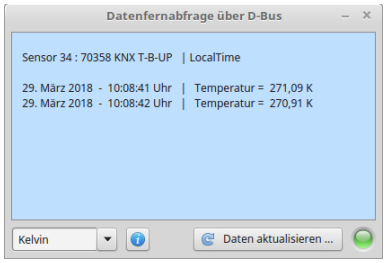

Abbildung 24.9.8.2.3: Zwei Temperatur-Abfragen

Über eine ComboBox können Sie das Argument für den Methodenaufruf festlegen, so dass die Temperaturwerte in der so festgelegten Temperaturskala abgerufen werden. Die Umrechnung erfolgt auf dem Server und ist Teil des angebotenen Services.

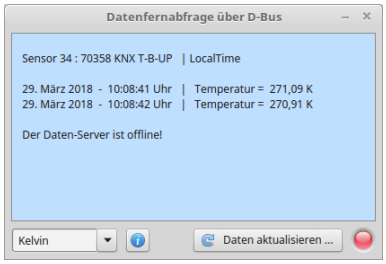

Abbildung 24.9.8.2.4: Der Server ist abgeschaltet

Über das Abschalten oder einen erfolgreichen Neustart des Servers werden Sie durch das permanente Überwachen des Signals "OnOff" durch den Client jederzeit optisch und akustisch informiert.

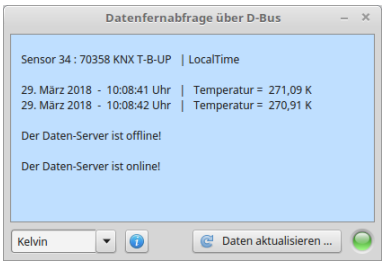

Abbildung 24.9.8.2.5: Der Server ist wieder online

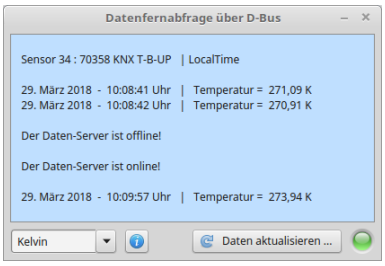

Abbildung 24.9.8.2.6: Der Service wird weiter genutzt ...

Jederzeit können Sie sich über die implementierte Methode, deren Signatur und den Eingabeparameter über eine implementierte Introspection informieren, die über einen Klick auf den i-Button realisiert wird. Das Fenster können Sie auch mit der ESC-Taste schließen.

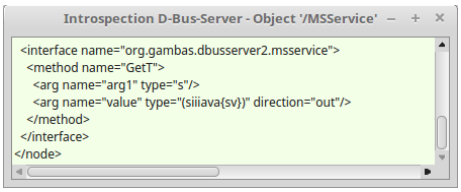

Abbildung 24.9.8.2.7: Erfolgreiche Introspection des Objektes "/MSService"

Das folgende Bild zeigt den Einsatz des Programm d-feet, um den Service des Servers zu nutzen:

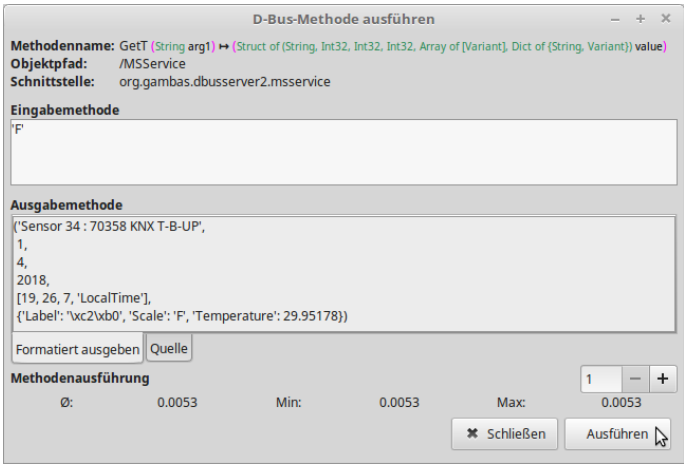

Abbildung 24.9.8.2.8: Methodenaufruf mit dem Argument 'F'