Institut für Betriebswirtschaftslehre und Wirtschaftsinformatik

D-Bus Language Bindings for ooRexx

The 2011 International Rexx Symposium

Rony G. Flatscher

Wirtschaftsuniversität Wien ■ Augasse 2-6 ■ A-1090 Wien

Agenda

- D-Bus
  - History
  - Usages
  - Concepts
- D-Bus Language Bindings for ooRexx ("dbusoorexx")
  - Overview
  - Examples
  - On-the-fly documentation
- Roundup and outlook

D-Bus
History

- History
  - RedHat, Inc.
    - Havoc Pennington
    - First release of the D-Bus specifications: 2003-09-06 (revision 0.8)
  - Handed over to "freedesktop.org"
    - Became part of all Linux distributions
  - Ported to other operating systems, e.g.
    - MacOSX
    - Windows

D-Bus
Usages, 1

- Linux kernel communicates with environment
  - Uses the "system" D-Bus daemon (a message broker)
  - Broadcasting D-Bus signals to report noteworthy events
    - E.g. reporting additions/removal of devices
  - For security reasons D-Bus services and interactions are controlled by system service configuration files
  - **Warning:** do *not change the service configuration files* with administrative privileges, if you are not 100% sure what you are doing!
    - You could harm your own system bad time!

D-Bus
Usages, 2

- Applications (services) within sessions
  - Uses the "session" D-Bus daemon (a message broker)
  - Using the user's credentials for using D-Bus services and interactions
  - Allows to interact with D-Bus "session" services using D-Bus messages
  - Allows to control the desktop and many applications
  - Allows to learn about events broadcasted as D-Bus signals from "session" services

D-Bus
Concepts, 1

- D-Bus Transports
  - Unix sockets, address prefix: "unix:"
    - Server and client on same computer
  - launchd, address prefix: "launchd:"
    - Server and client on same computer
  - nonce-TCP/IP sockets, address prefix: "nonce-tcp:"
    - Server and client on same computer
  - TCP/IP sockets, address prefix: "tcp:"
    - Server and client on same *or different* computer

D-Bus
Concepts, 2

- D-Bus Messages
  - Employing a transport, D-Bus messages can be exchanged
  - Message consists of an interface name and a member name
  - There are four message types
    - "call message" that may cause a "reply message" or an "error message" (or no reply at all)
    - a one-way "signal message"
  - Arguments and return values are strictly typed
    - 13 basic types (boolean, byte, double, int16, float, string, …)
    - 4 container types (array, map/dict, structure, variant)

D-Bus Datatypes

array	**a**	.Array
boolean	**b**	Rexx string
byte	**y**	Rexx string
double	**d**	Rexx string
int16	**n**	Rexx string
int32	**i**	Rexx string
in64	**x**	Rexx string
objpath	**o**	Rexx string
signature	**g**	Rexx string
string	**s**	Rexx string
uint16	**q**	Rexx string
uint32	**u**	Rexx string
uint64	**t**	Rexx string
unix_fd	**h**	Rexx string
variant	**v**	depends on signature
structure	**( )**	.Array
map / dict	**a{s…}**	.Directory

Some examples:

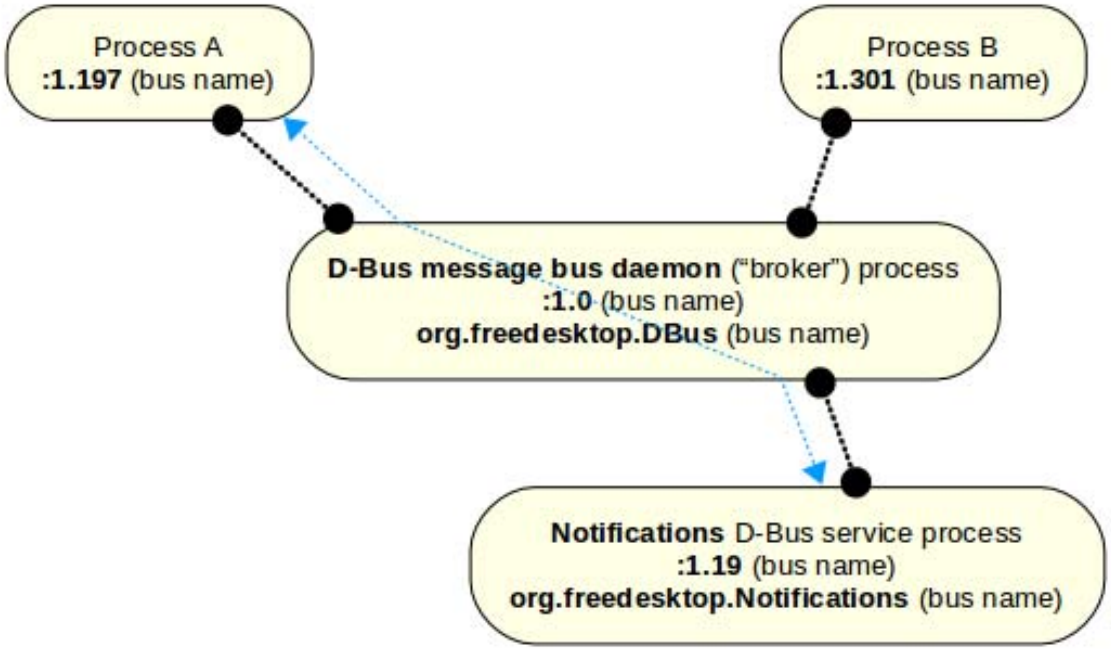
```
org.freedesktop.DBus.Introspectable
s       Introspect()

org.freedesktop.DBus.Properties
v       Get(ss)
a{sv}   GetAll(s)
        Set(ssv)

org.freedesktop.DBus.Notifications
        CloseNotification(u)
as      GetCapabilities()
(ssss)  GetServerInformation()
u       Notify(susssasa{sv}i)
```

…

D-Bus
Concepts, 3

- D-Bus Connection
  - A connection between a D-Bus client and a D-Bus server
  - Dubbed "bus"
- D-Bus Message Daemon/Broker
  - A D-Bus server
  - A set of services that allow it to act as a message broker
    - Relays D-Bus messages among D-Bus clients connected to it
  - Manages D-Bus connections
    - Allows to assign one or more unique names to connections
  - Can start D-Bus services on demand

D-Bus
D-Bus Message Daemon / Broker



D-Bus
Concepts, 4

- Object Path
  - A String starting with "/"
  - Denotes the "object" one wishes to send a D-Bus message to
- Sending D-Bus messages
  - Unique bus name, service name
  - Object path
  - Interface name
  - Member name
    - Arguments

D-Bus
Concepts, 5

- Discovering D-Bus service object interfaces on the fly
  - Message org.freedesktop.DBus.Introspectable.Introspect()
    - Returns a XML-encoded file with the interface definitions
  - Addressed to a D-Bus object in a D-Bus service

D-Bus
Concepts, 6

- Private D-Bus Server
  - Allows to create a simple "private" D-Bus server
    - No daemon/broker services available
  - D-Bus clients can interact with D-Bus server
    - D-Bus infrastructure allows to
      - Connect to a (private) D-Bus server
      - Exchange D-Bus messages with the D-Bus server
  - Makes it easy to create client-server apps fast
    - If using the tcp-transport, then D-Bus based interactions can be accross multiple computers!

D-Bus Language Bindings for ooRexx

(Beta as of December 2011, hence details may change)

D-Bus Language Bindings for ooRexx
Download and Installation

- Download (beta)
  - http://wi.wu.ac.at/rgf/tmp/dbus/onthefly/
  - Please report errors and ask questions on the <news:comp.lang.rexx> newsgroup
- Installation (currently Linux only)
  - rexx install_ooRexx_dbus.rex
- Uninstall (currently Linux only)
  - rexx install_ooRexx_dbus.rex -u

D-Bus Language Bindings for ooRexx Overview, 1

- Combination of native code ("dbusoorexx") and the ooRexx package named "dbus.cls"
  - Closely coupled
    - "dbusoorexx" depends on classes and behaviour of "dbus.cls"
    - "dbus.cls" depends on the features and behaviour of "dbusoorexx"

    → Do not change the code, unless you know what you are doing!
  - Goals
    - Make it easy for ooRexx programmers to interact with D-Bus
      - Take advantage of a dynamically typed language
      - Apply the Rexx "human-orientation" philosophy where possible

D-Bus Language Bindings for ooRexx Overview, 2

- "dbus.cls"
  - Defines ooRexx classes for the D-Bus language binding
    - DBus
      - Core class to allow
        - Connecting to D-Bus daemons (e.g. "system", "session", address)
        - Sending distinct call and signal messages to D-Bus services
        - Filtering and fetching signal messages from other D-Bus services
        - Getting ooRexx proxy objects for D-Bus service objects
    - DBusProxy
      - Utility class to camouflage a service object as an ooRexx object
        - Returned by .DBus method getObject(busName,objectPath)
      - Automatic method lookup, marshalling of arguments and unmarshalling of return values

D-Bus Language Bindings for ooRexx Overview, 3

- DBusServiceObject
  - Allows ooRexx objects to be used as D-Bus service objects
- DBusSignalListener
  - Implicitly used by .DBus
  - Allows for additional filtering of D-Bus signal messages
- DBusServer
  - Allows to create a private D-Bus server in ooRexx

D-Bus Language Bindings for ooRexx Overview, 4

- IDBus, IDBusNode, IDBusInterface, IDBusMethod, IDBusCallMethod, IDBusSignalMethod, IDBusPropertyMethod, IDBusArg, IDBusAnnotation
  - Utility classes for introspection of D-Bus service objects
  - Needed by classes and routines in "dbus.cls"
  - Usually not used by ooRexx programmers
- IntrospectHelper, IntrospectHelperInterface
  - Utility classes to create introspection data on-the-fly
- IDBusPathMaker
  - Utility class to set up D-Bus service-object discovery for ooRexx DBusServiceObjects

D-Bus Language Bindings for ooRexx Overview, 5

- Public routines
  - dbus.box(signature[,args])
    - Needed for variant values that expect a specific signature
  - string2UTF8(string)
    - D-Bus string datatype must be UTF-8
    - Converts a Rexx string to UTF-8 (if it contains non-US characters)
  - DBusDataType(value[,type])
    - Returns the D-Bus datatype name of value, else .nil
    - If type argument given, returns .true or .false, type can be:
      - B[usname], I[nterfaceName] , M[ember], O[bjectPath], S[ignature]

D-Bus Language Bindings for ooRexx Examples, 1

- Using a common service
  - Bus name ("service name")

    org.freedesktop.Notifications
  - Object path

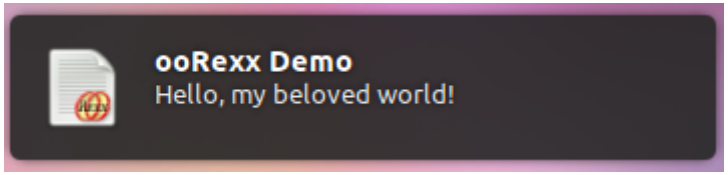
    /org/freedesktop/Notifications
  - Interface name

    org.freedesktop.Notifications
    - Members

```
                CloseNotification(u)
as              GetCapabilities()
(ssss)          GetServerInformation()
u               Notify(susssasa{sv}i)
```

D-Bus Language Bindings for ooRexx Examples, 2

```
conn=.dbus~session      /* get connection to session dbus       */

    /* define message arguments */
busName       ="org.freedesktop.Notifications"
objectName    ="/org/freedesktop/Notifications"
interfaceName ="org.freedesktop.Notifications"
memberName    ="Notify"
replySignature="u"             /* uint32 */
callSignature ="susssasa{sv}i" /* string,uint32,string,string,string,array of string,dict,int32 */

id=conn~message("call",busName,objectName,interfaceName,memberName,replySignature,callSignature, -
               "An ooRexx App", , "oorexx", "ooRexx Demo", "Hello, my beloved world!", , , -1)

::requires "dbus.cls"          /* get DBus support  */
```



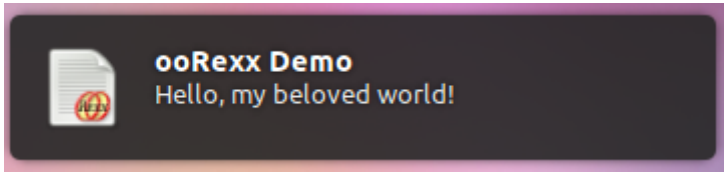
D-Bus Language Bindings for ooRexx Examples, 3

- Getting the D-Bus service object as an ooRexx object
  - .DBus method getObject(busName,objectPath)
  - returns a DBusProxyObject which
    - Remembers the bus name and the object path
      - Used for sending messages
    - Interrogates the interfaces of the target D-Bus service object
      - Used for automatically determining methods, marshalling arguments and unmarshalling return values

  → Very simple and easy to interact with D-Bus service objects!

D-Bus Language Bindings for ooRexx Examples, 4

```
 /* get access to remote object */
o=.dbus~session~getObject("org.freedesktop.Notifications","/org/freedesktop/Notifications")

id=o~notify("An ooRexx App", , "oorexx", "ooRexx Demo", "Hello, my beloved world!", , , -1)

::requires "dbus.cls"      /* get DBus support  */
```

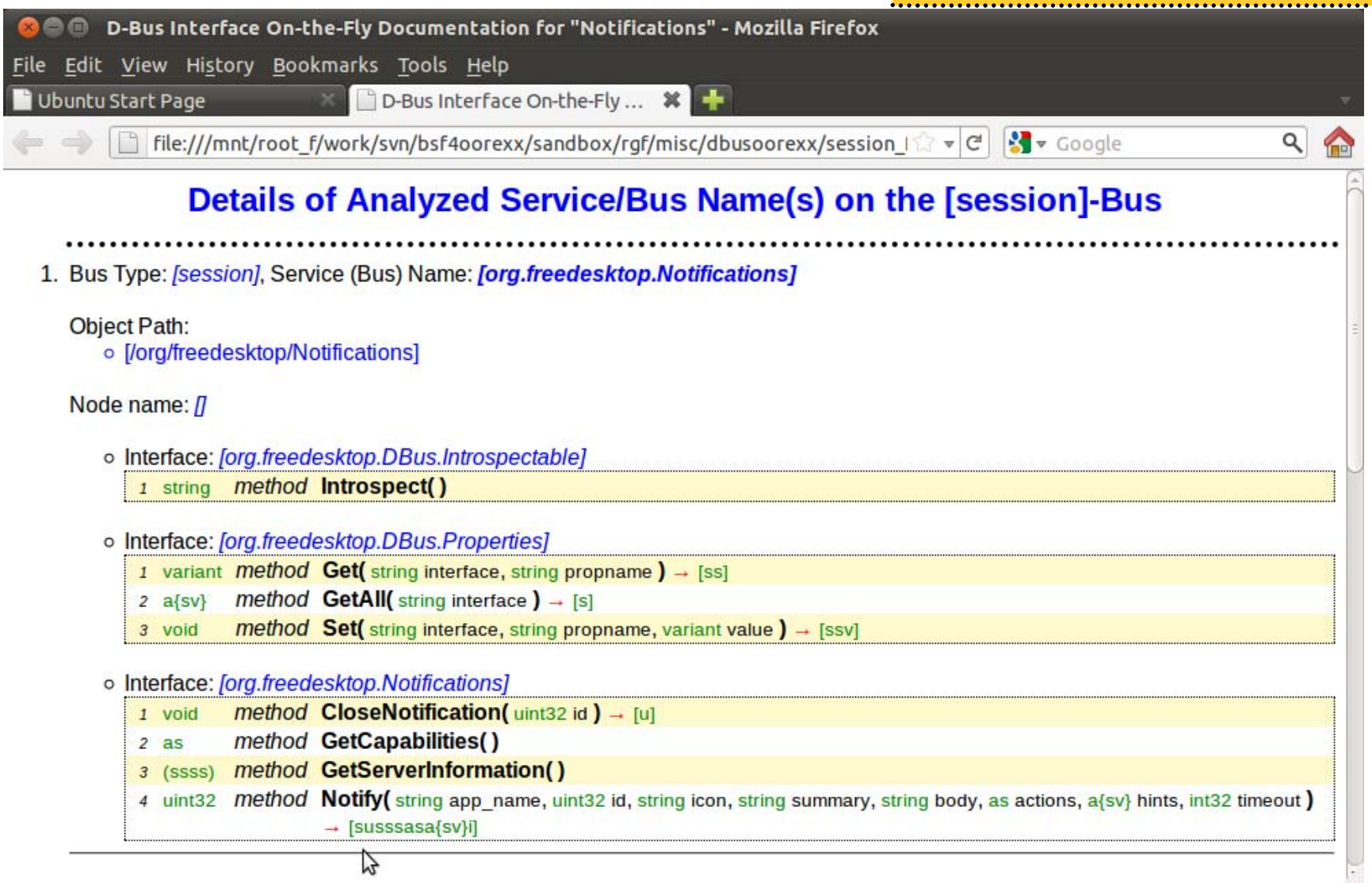


D-Bus Language Bindings for ooRexx
On-the-fly Documentation, 1

- D-Bus documentation sometimes "meager"
- Idea to exploit the D-Bus infrastructure
  - The "org.freedesktop.DBus" family of interfaces
  - org.freedesktop.DBus.Introspection.Introspect()
    - Usually implemented by every D-Bus service objects
- Render interface definitions as HTML text
  - Format results with CSS to allow easy usage, format changes
  - Collect complex signatures and list them at the end

D-Bus Language Bindings for ooRexx
On-the-fly Documentation, 2

```
rexx dbusdoc.rex Notifications
```



A Side-note on NetRexx

- NetRexx needs to use the Java language bindings of D-Bus
  - Java implementation independent from the C-based implementations
  - Java programmer is expected to create and compile D-Bus service related interface classes
    - Utilities to create the respective Java interface skeletons
    - Service object's interfaces may be different on different platforms!
      - Compiled variants needed for different platforms and service implementations!
  - Only "fossilized" implementations possible in Java, hence in NetRexx!
- NetRexx may exploit the flexible, dynamic ooRexx D-Bus!
  - BSF4ooRexx framework

→Fast and easy execution of [oo]Rexx D-Bus scripts !

Roundup and Outlook

- Genuine ooRexx language binding for ooRexx
  - 32- and 64-bit ports available
  - Deployable on all Linux systems
- Makes it very easy to exploit D-Bus
  - Rexx philosophy "human-orientness" a guiding principle
  - All D-Bus service objects can be interacted with
  - All D-Bus signals (events) can be handled
- ooRexx D-Bus service objects easy to implement!
- Beta version: http://wi.wu.ac.at/rgf/tmp/dbus/onthefly/
- Support for other D-Bus platforms coming up
  - MacOSX
  - Windows