D-Bus bindings for Tcl

Inter-process communication
with Tcl on Linux

D-Bus introduction

- Available on all reasonably current versions of Linux
- Default system and session busses
- Additional busses can be started, if desired
- Applications can:
  - Report events using D-Bus signals
  - Interact with each other via D-Bus method calls
  - Register a well-known name on the D-Bus
    - Makes them easy to locate by other applications
    - Can be used to allow only one instance of an application
  - Be started automatically on demand
- Standard interfaces provide additional functionality:
  - Introspection
  - Properties
- Organizes items of an application by path and interface

Introspection

```
# qdbus.tcl org.freedesktop.ScreenSaver /ScreenSaver
signal void org.freedesktop.ScreenSaver.ActiveChanged(bool)
method bool org.freedesktop.ScreenSaver.GetActive()
method uint org.freedesktop.ScreenSaver.GetActiveTime()
method bool org.freedesktop.ScreenSaver.SetActive(bool e)
          :
method QDBusVariant org.freedesktop.DBus.Properties.Get(QString
   interface_name, QString property_name)
method QVariantMap org.freedesktop.DBus.Properties.GetAll(QString
   interface_name)
method void org.freedesktop.DBus.Properties.Set(QString
   interface_name, QString property_name, QDBusVariant value)
method QString org.freedesktop.DBus.Introspectable.Introspect()
method QString org.freedesktop.DBus.Peer.GetMachineId()
method void org.freedesktop.DBus.Peer.Ping()
```

Tcl dbus library

```
package require dbus
dbus connect
dbus info name

dbus call -dest [dbus info service] \
    [dbus info path] [dbus info service] ListNames
dbus call -dest org.freedesktop.ScreenSaver \
    /ScreenSaver org.freedesktop.ScreenSaver GetActive
dbus call -dest org.freedesktop.ScreenSaver -signature b \
    /ScreenSaver org.freedesktop.ScreenSaver SetActive 1

dbus method /debug com.tclcode.debug.Eval {
    apply {{info str} {uplevel #0 $str}}
}

dbus signal /debug error $errorInfo
```

Tcl dbus interface

```
package require dbif

dbif connect com.tclcode.demo
dbif method -interface com.tclcode.debug \
    /debug Eval str result {uplevel #0 $str}
dbif signal -id logentry /debug log message:xss {level str} {
    return [list [clock milliseconds] $level $msg]
}
dbif property -access read /debug version tcl_version

dbif generate logentry error $errorInfo

# qdbus.tcl com.tclcode.demo /debug
property read QString com.tclcode.demo.version
signal void com.tclcode.demo.log(QDBusRawType::xss message)
method QString com.tclcode.debug.Eval(QString str)
```

Advanced property use

```
package require dbif

dbif connect com.tclcode.demo

dbif property -attributes {Property.EmitsChangedSignal false} \
      / pwd cwd {cd $pwd}

trace add variable cwd read {
      apply {args {set ::cwd [pwd]}}
}
```

Advanced method use

```
dbif method / CheckUpdate {} version {
    global updateurl
    http::geturl $updateurl -command [list checkupdate $msgid]
    return -async 1
}

proc checkupdate {msgid token} {
    if {[http::status $token] ne ”ok” || [http::ncode $token] ne "200"} {
        dbif error $msgid "Update check failed"
    } else {
        dbif return $msgid [http::data $token]
    }
    http::cleanup $token
}
```