Introduction à DBus

DBus est un système de communication entre processus mis au point par *RedHat*, et aujourd'hui utilisé dans toutes les distributions GNU/Linux. Il permet à la fois d'interroger les paramètres de configuration du système, d'être prévenu de l'arrivée ou du départ de nouveaux périphériques, de faire de la surveillance en temps réel (monitoring), etc.

Pour qui veut développer des application interactives, avec le noyau ou le reste du système, c'est clairement la brique logicielle à connaître et utiliser, sachant que tous les langages de programmation majeurs proposent aujourd'hui des librairies **DBus** prêtent à l'emploi...

Nous nous restreindrons ici à une découverte rapide depuis la ligne de commande, via l'outil `busctl`, proposé dans **Systemd** par défaut. Une bonne habitude à prendre en développement est d'utiliser `dbus monitor NOM_DU_SERVICE` pour traquer les messages spécifiques à un objet **DBus** donné.

Côté interface graphique, on trouve `d-feet`, qui ne brille peut-être pas par son ergonomie, mais qui fait parfaitement son boulot !

Légende des couleurs utilisées dans ce document :

➔ Nom du service/Destination Chemin/Path Interface *Method* Arguments

➔ Signal

Interfaces génériques

Par défaut, quel que soit l'objet **DBus** étudié, on retrouve les interfaces :

➔ org.freedesktop.DBus.Properties avec les méthodes *GetAll / Get / Set* pour lire/écrire dans les propriétés de l'objet, et le signal PropertiesChanged pour être avertit quand le noyau change des propriétés

➔ org.freedesktop.Dbus.Peer avec les méthodes *GetMachineId / Ping*

➔ org.freedesktop.Dbus.Introspectable avec sa méthode *Introspect* pour retrouver la syntaxe complète des autres méthodes et avoir rapidement un aperçu des propriétés de l'objet.

Étude du gestionnaire réseau (network-manager)

Arborescence générale du service

```
busctl tree org.freedesktop.NetworkManager
```

Dans la suite, on s'intéressera au chemin gérant les réglages (*Settings*) présents.

Recherche de tous les éléments internes du service

```
busctl introspect org.freedesktop.NetworkManager /org/freedesktop/NetworkManager/Settings
```

Extrait des propriétés des Settings (réglages) au format JSON

```
busctl -j call org.freedesktop.NetworkManager /org/freedesktop/NetworkManager/Settings org.freedesktop.DBus.Properties GetAll s
org.freedesktop.NetworkManager.Settings
```

Extrait d'une propriété des Settings au format JSON

```
busctl -j call org.freedesktop.NetworkManager /org/freedesktop/NetworkManager/Settings org.freedesktop.DBus.Properties Get ss
org.freedesktop.NetworkManager.Settings Hostname
```

Extrait de la liste des connexions au format JSON

```
busctl -j call org.freedesktop.NetworkManager /org/freedesktop/NetworkManager/Settings org.freedesktop.NetworkManager.Settings ListConnections
```

Propriétés de la connexion 1 au format JSON

```
busctl -j call org.freedesktop.NetworkManager /org/freedesktop/NetworkManager/ActiveConnection/1 org.freedesktop.DBus.Properties GetAll s org.freedesktop.NetworkManager.Connection.Active
```

Introspection des réglages de cette connexion

```
busctl introspect org.freedesktop.NetworkManager /org/freedesktop/NetworkManager/Settings/1
```

Affichage des propriétés des réglages de cette connexion

```
busctl -j call org.freedesktop.NetworkManager /org/freedesktop/NetworkManager/Settings/1 org.freedesktop.DBus.Properties GetAll s org.freedesktop.NetworkManager.Settings.Connection
```

Introspection du périphérique de cette connexion

```
busctl introspect org.freedesktop.NetworkManager /org/freedesktop/NetworkManager/Devices/1
```

Affichage des propriétés de cette connexion

Attention : les interfaces suivantes peuvent varier suivant votre configuration réseau...

```
busctl -j call org.freedesktop.NetworkManager /org/freedesktop/NetworkManager/Devices/1 org.freedesktop.DBus.Properties GetAll s org.freedesktop.NetworkManager.Device
busctl -j call org.freedesktop.NetworkManager /org/freedesktop/NetworkManager/Devices/1 org.freedesktop.DBus.Properties GetAll s org.freedesktop.NetworkManager.Device.Statistics
busctl -j call org.freedesktop.NetworkManager /org/freedesktop/NetworkManager/Devices/1 org.freedesktop.DBus.Properties GetAll s org.freedesktop.NetworkManager.Device.Wired
```

Introspection de la connexion active

```
busctl introspect org.freedesktop.NetworkManager /org/freedesktop/NetworkManager/ActiveConnection/1
```

Autres exemples hors réseau

Afficher le niveau de deboguage de Systemd

On utilisera ici les arguments get-property et set-property qui offrent des raccourcis à l'interface org.freedesktop.DBus.Properties...

```
busctl get-property org.freedesktop.systemd1 /org/freedesktop/systemd1 org.freedesktop.systemd1.Manager LogLevel
```

Modifier ce niveau en debug

```
busctl set-property org.freedesktop.systemd1 /org/freedesktop/systemd1 org.freedesktop.systemd1.Manager LogLevel s debug
```

Variables d'environnement sans Systemd

```
busctl get-property org.freedesktop.systemd1 /org/freedesktop/systemd1 org.freedesktop.systemd1.Manager Environment
```

Lancer l'unité Systemd de CUPS (le serveur d'impression)

```
busctl call org.freedesktop.systemd1 /org/freedesktop/systemd1 org.freedesktop.systemd1.Manager StartUnit ss "cups.service" "replace"
```

Conclusion

La logique globale de DBus est en fait assez simple, une fois qu'on a compris le rôle des interfaces et méthodes associées.

En revanche, on aurait aimé pouvoir éviter à chaque fois l'écriture du `org.freedesktop`, et il faut également reconnaître que la syntaxe des arguments est parfois assez déroutante, avec de légères variations suivant les langages de programmation.

C'est souvent ce dernier élément qui fait perdre beaucoup de temps en développement, à devoir tester et trouver la bonne syntaxe dans la « signature », qui décrit le type des arguments en envoi, puis dans l'« objet » d'envoi, qui contient les données à fournir à une méthode.