S. Lavirotte – J.-Y. Tigli
Polytech'Nice – Sophia
SI5
2012-2013

TD
UPnP: Universal Plug and Play

1 Les outils INTEL pour la Technologie UPnP

Ce TD est largement inspiré des tutoriaux vidéo d'Intel que vous pourrez trouver dans

http://kistren.polytech.unice.fr/cours/oc/td1/videos/

1. Télécharger et installer les outils **Developer Tools for UPnP Technologies** version 0.0.56

http://kistren.polytech.unice.fr/cours/oc/td1/DeveloperToolsForUPnPTechnologies.msi

ou pour la dernière version sur le site d'OpenTools UPnP :

http://opentools.homeip.net/dev-tools-for-upnp

2. Lancer l'outil « *Device Spy* », qui le un point de contrôle universel d'Intel appelé UCP (*Universal Control Point*). Cet outil permet de tester des invocations de commande et des réceptions d'événements depuis des dispositifs UPnP. Lancer le dispositif UPnP « *Network Light* », vérifier son apparition et tester sa manipulation depuis UCP.

2 Développements UPnP à l'aide d'Outils

Nous vous encourageons à visualiser la vidéo « Overview of Intel® Device Builder, part of the Intel Authoring Tools for UPnP Technologies » avant de commencer ce TD si vous utilisez ces outils afin de générer le code squelette de votre dispositif UPnP.

2.1 Dispositif Switch

Utiliser l'outil **Service Author** pour décrire interactivement le service de votre premier dispositif UPnP : un dispositif de type Interrupteur (*Switch*).

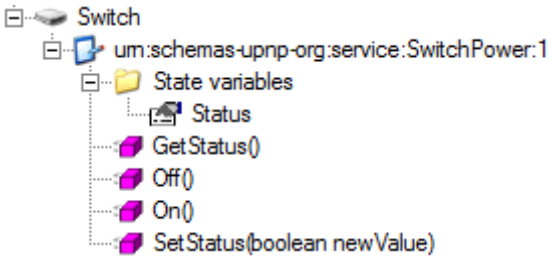

Figure 1: Dispositif "Interrupteur Virtuel"

2.1.1 Création du Service pour notre dispositif Switch

Commencez par créer le service `SwitchPower` (voir la description du service du dispositif sur la figure ci-dessus). Vous veillerez à déclarer le variable booléenne `Status` en variable événementielle. Bien entendu, la méthode `GetStatus` retournera la valeur de `Status` (de type boolean) et la méthode SetStatus prendra an paramètre un boolean. On pourra ensuite visualiser le fichier XML généré pour comprendre l'intérêt de l'outil.

2.1.2 Création du Dispositif Switch par assemblage de Services

Utiliser le **Device Builder** pour créer votre dispositif et y attacher le service que vous venez de créer (inclure le fichier généré par le **Service Author**).

2.1.3 Génération du code pour le dispositif Switch

Device Builder vous permet de générer le squelette de code correspondant à votre dispositif « *Switch* ». Vous veillerez à choisir la génération de code pour l'environnement Visual Studio .Net C#. Vous veillerez à modifier le namespace du code généré au nom : « Switch ».

2.2 Dispositif Light

Nous allons maintenant tenter de réécrire le programme « *Network Light* ».

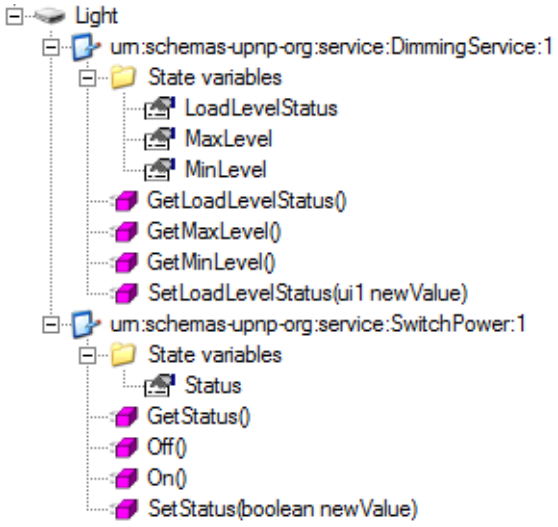

Figure 2: Description du dispositif "Lampe Virtuelle"

2.2.1 Services pour notre dispositif Light

Créer le service `DimmingService` (voir la description du service du dispositif sur la figure ci-dessus). Nous réutiliserons le service `SwitchPower` que nous avons déjà réalisé pour l'interrupteur car il est bien identique à celui-ci. Le nouveau service doit permettre de contrôler la luminosité du dispositif. Le niveau de luminosité devra être exprimé en pourcentage (entre 0 et 100 avec un pas de 10), le niveau maximum étant 100 et le minimum 0. La fonction `GetLoadLevelStatus` devra retourner le niveau de luminosité.

2.2.2 Génération du code pour le dispositif Light

Vous répéterez les étapes décrites dans les sections 2.1.2 et 2.1.3 afin de produire le squelette du dispositif *Light*.

2.3 Implémentation du comportement du Dispositif

2.3.1 Version Textuelle du Dispositif

Le générateur de code a créé plusieurs classes. Etudier plus particulièrement le contenu du fichier `SampleDevice.cs` (code correspondant au dispositif) et les deux fichiers créés pour chacun des services (`DimmingService.cs` et `SwitchPower.cs`).
Modifier le code pour implémenter les actions associées à votre dispositif. De simples messages dans la console textes permettront de vérifier le comportement correct de votre dispositif.

2.3.2 Version graphique du Dispositif

Pour avoir une interface graphique pour vos dispositifs, utilisez le projet fourni à l'adresse suivante :

`http://kistren.polytech.unice.fr/cours/oc/td1/Templates.zip`

et remplacez les fichiers `SamplesDevice.cs`, `DimingService.cs` et `SwitchPower.cs` que vous avez obtenus lors de la version précédente. Il ne vous reste plus qu'à faire les appels aux méthodes graphiques au lieu des affichages textuels dans la console.

2.4 Développement de Point de Contrôle UPnP

Utiliser le **Device Builder** pour générer le code en C# sous Visual Studio .Net du point de contrôle associée au dispositif UPnP « *Network Light* » des outils Intel ou à votre propre dispositif que vous venez de réaliser. Modifier

le code pour implémenter allumer et éteindre la lampe depuis votre code du point de contrôle. Vous venez de réaliser un **point de contrôle de type interrupteur** (à ne pas confondre avec le **dispositif interrupteur** que nous avons créé à la section 2.1 qui ne permet pour le moment pas de contrôler quoi que ce soit).

3 Développement UPnP sans Outils

L'utilisation des outils présentés précédemment n'est pas obligatoire. Vous pouvez aussi directement écrire votre code en C# par exemple. Dans le cas de dispositifs simples comme ceux que nous avons décrits dans la section précédente, il est tout à fait possible de les implémenter sans ces outils. La documentation sur la pile UPnP que nous avons récupérée n'est pas écrite. Mais vous pourrez vous référer au code source qui est téléchargeable ou bien en décompilant la dll.

Pour la création d'un dispositif, vous aurez besoin de rootDevice = UPnPDevice.**CreateRootDevice** puis de modifier les valeurs de l'objet retourné par cette méthode pour modifier le `FriendlyName`, le `DeviceURN`, …

Pour la création d'un service vous utiliserez listServices = **new UPnPService**, puis vous ajouterez les variables d'état grâce à listServices.**AddStateVariable** et les méthodes grâce à listServices.**AddMethod** . Le rajout d'un service à un device s'effectue grâce à rootDevice.**AddService**. Il ne reste alors plus qu'à faire appel dans le constructeur à rootDevice.**Advertise**().

Le démarrage et l'arrêt du dispositif se font respectivement grâce à rootDevice.**StartDevice**() et rootDevice.**StopDevice**().

4 Mise en œuvre de l'audio et vidéo avec UPnP

La seconde manipulation consiste à tester les éléments de l'architecture AV (Audio/Vidéo) d'UPnP. Vous pourrez vous reporter au tutorial vidéo d'Intel : **Overview of Intel® AV Tools for UPnP Technologies.**

Le kit d'Intel comporte :

- un dispositif AV Media Server qui permet de servir (en streaming) des sources audio et vidéo,
- un dispositif AV Media Renderer qui permet de jouer des sources audio et vidéo,
- deux points de contrôle AV Wizard et AV Media Controller qui permettent respectivement de piloter (télécommande) le programme AV Media Renderer et de lister des sources audio et vidéo disponibles sur les serveurs et de piloter leurs lectures sur les renderers.

1. Lancez ces 4 éléments dans l'ordre suivante : AV Wizard, AV Media Controller, AV Media Renderer et AV Media Server.
2. Explorez les services fournis par le(s) serveur(s),
3. Publiez des fichiers multimédia (audio ou vidéo) sur votre serveur,
4. Démarrez la lecture d'une vidéo sur votre Renderer au moyen des points de contrôle ouverts sur votre poste, mais aussi éventuellement, prendre la main sur le Renderer d'une autre machine.
5. S'il vous reste du temps, vous pouvez tenter de développer votre propre application de pilotage des AV Media Server et AV Media Renderer.

5 GNU/Linux et UPnP

Les inconditionnels de Linux (et pour ceux qui ont déjà implémenté des dispositifs sous Windows) pourront s'essayer dans l'utilisation du SDK Linux:

http://pupnp.sourceforge.net/

et vérifiez ainsi l'interopérabilité d'UPnP entre différents OS.