**Accès aux équipements**

**-**

**Base Drivers**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Confidentialité : consortium** | | | | | | |
| **Historique des versions successives** | | | | | | |
| **INDICE** | **DATE** | **Commentaires** | | | **Redac.** | **Relect.** |
| **v0.0** | **28/11/2013** |  | | | **J. Pulou** |  |
| **V0.1** | **29/11/2013** |  | | | **A. Bottaro** |  |
| **V0.2** | **2/11/2013** |  | | | **J.Pulou** | **Orange** |
| **Document établi sous la responsabilité de** | | | | | | |
| **Rédaction** | | | **Relecture** | **Approbation par le consortium** | | |
| **Nom : Jacques Pulou, Pierre Rust, André Bottaro, Gregory Bonnardel** | | | **Nom :** | **Date :** | | |
| **Date : 29/11/2013** | | | **Date :** |
| **Résumé :**  **Le projet Open The Box vise à la définition, au sein d'un consortium d'entreprise, d'une infrastructure ouverte au déploiement d’applications dans le réseau domestique sur les marchés de la Maison Intelligente. Dans l’architecture technique du projet, les applications peuvent accéder aux équipements domestiques au travers d’APIs mise à disposition à distance (cf. livrable 4.3.2) ou localement dans un langage de programmation Java sur l’environnement d’exécution OSGi embarqué.**  **Les APIs locales d’accès aux équipements sont implémentées par des pilotes appelés ‘base drivers’. Dans le cadre du projet Open the Box, des base drivers sont fournis pour les trois technologies suivantes : ZigBee, X3D, GreenNet. Ce document décrit les principes de ces base drivers et présente les caractéristiques communes requises par l’infrastructure Open the Box pour. Ces caractéristiques sont demandées par les fonctionnalités d’administration par l’utilisateur (dashboard), d’administration par l’opérateur (infrastructure TR-069) et d’accès distant.** | | | | | | |

Table des matières

[I. Avertissement 4](#_Toc373759919)

[II. Introduction 4](#_Toc373759920)

[III. Plug‘n’Play 4](#_Toc373759921)

[API et Pattern de programmation 4](#_Toc373759922)

[IV. Propriétés attachées à un service 5](#_Toc373759923)

[V. References 6](#_Toc373759924)

# Avertissement

La spécification de référence pour l’implémentation des « base drivers » se trouve dans la spécification OSGi device\_Access. Le présent document n’est qu’un guide destiné à faciliter la vie aux développeurs et à préciser certains points spécifiques à OTB. Ce document sera aussi amené à évoluer dans le temps en fonctions des retours et des besoins identifiés dans les phases de développement, intégration et tests.

# Introduction

Les Base drivers ont pour fonction de réifier les services apportés par les équipements connectés à la HAB (Home Automation Box) afin de les exposer sous forme de services OSGi et de les rendre accessibles pour le programmeur d’application métier.

Dans l’architecture logicielle retenue pour le projet Open the Box, il existe autant de Base Drivers qu’il existe de technologies de connexion.

Pour mémoire, vous trouverez ci-dessous les Base Drivers développés dans le cadre du projet Open the Box.

|  |  |  |  |
| --- | --- | --- | --- |
| Technologies | Auteurs | Licence | Site de Dépôt ou personne à solliciter |
| ZigBee | Orange | Orange | <https://minalogic.net/file/showfiles.php?group_id=169> |
| X3D | Delta Dore | Delta Dore |  |
| GreenNet | STM | STM |  |
| USB | Bouygues Telecom | Bouygues Telecom |  |
|  |  |  |  |
|  |  |  |  |

# Plug‘n’Play

Le Base Driver est intégré dans la mise en œuvre de la fonctionnalité Plug’n’Play. Une fois l’appairage réalisé (connectivité couche basse niveau physique), le Base Driver prend en charge de façon automatique (sans intervention de l’utilisateur) la réification des services offerts par l’équipement. Un base driver doit être « auto-porteur ». En particulier il doit publier les services des devices qu’il supporte indépendamment de la présence, ou non, d’une application les utilisant.

## API et Pattern de programmation

Chaque service offert par l’équipement (certains équipements peuvent être multi-services suivant les technologies) est réifié sous la forme d’un objet OSGi (proxy) implémentant une interface partagée (e..g, ZigBeeEndPoint pour la technologie ZigBee) et publié en tant que service dans le registre de services OSGi. Une fois les services enregistrés, les bundles applicatifs peuvent requérir ces services auprès du registre à la condition qu’ils disposent des permissions éventuellement nécessaires. Ils peuvent alors invoquer des méthodes sur ces services que le Base Driver traduit sur le réseau par les messages appropriés.

# Propriétés attachées à un service

Lors de la publication d'un service OSGi représentant un équipement, Le base driver correspondant à la technologie de connexion utilisée doit lui attacher des propriétés suivantes :

|  |  |  |  |
| --- | --- | --- | --- |
| Ident | Status | Type | Comment & Reference |
| Connectivity independent properties | | | |
| DEVICE\_CATEGORY | MUST | String[]. Le premier élément du tableau prenant comme valeurs : X3D,ZiigBee,USB,UPnP… | Connectivity technology [1], could be completed as needed. |
| DEVICE\_DESCRIPTION | SHOULD | String. Description d’équipement selon la spécification OSGi Device Access [1] |  |
| DEVICE\_SERIAL | SHOULD | String. Numéro de série de l’équipement selon la spécification OSGi Device Access [1] | OSGi Device Access [1] |
| service.pid | MUST | String. Un identifiant unique selon la spécification OSGi Device Access [1] | service Persistent ID[1] |
| DEVICE\_FRIENDLY\_NAME | MUST | String. Un nom d’équipement en anglais d’un ou plusieurs mots et de moins de 25 caractères. | si cette propriété n'est pas présente, les 25 premiers caractères de DEVICE\_DESCRIPTION seront utilisés pour afficher l’équipement dans l'interface utilisateur |
| Connectivity dependent properties (zigBee example hereafter) | | | |
| NODE\_TYPE |  |  | ZigBee exemple [2] |
| MANUFACTURER\_CODE |  |  | ZigBee exemple [2] |
| HOST\_PID |  |  | ZigBee exemple [2] |

**Accès aux caractéristiques des équipements et invocation de commandes ?**

* Des interfaces spécifiques à chaque technologie (ZigBee, X3D, GreenNet) représentent les concepts du modèle spécifié. Par exemple, des interfaces ZigBeeNode, ZigBeeEndPoint, ZigBeeCluster, ZigBeeCommand représente sous forme objet les concepts node, endpoint, cluster, command de la spécification ZigBee. Le modèle peut être plus simple comme pour la technologie X3D où seule l’interface X3DDevice est spécifiée. Ces interfaces permettent généralement d’énumérer les caractéristiques et les commandes découvrables sur les équipements selon la technologie associée.
* Après avoir découvert dans le registre OSGi les objets représentant les équipements, une application peut invoquer des commandes par appels de méthodes sur l’interface pertinente, par exemple ZigBeeCommand.invoke(byte[], ZigBeeHandler) pour ZigBee et X3DDevice.executeCommand(String cmd, String[] arguments, X3DHandler) pour X3D.
* L’invocation de commandes et la réception d’une éventuelle réponse sont asynchrones. L’invocation de commandes est par conséquent réifié avec un ‘handler’ parmi les arguments. Le handler est un objet construit par l’application que le base driver notifie de manière asynchrone dès que la réponse est disponible.

**Notification d’événements**

Les protocoles tels que ZigBee, X3D, GreenNet permettent la notification d’événements. Afin de souscrire à la notification d’événements, chaque application enregistre un service d’écoute d’événements, c’est-à-dire un ‘listener’. Ce service implémente une interface appelée par exemple « ZigBeeEventListener » dans le cas de ZigBee, ou appelé EventHandler dans le cas du base driver X3D qui utilise le service standard Event Admin. L’application enregistre ce service avec des propriétés qualifiant le type d’événement souscrit (par exemple l’identifiant de l’équipement visé). Le base driver, par utilisation du registre de services, prend connaissance de ces souscriptions et les traduit dans les mécanismes protocolaires de la technologie associée. Dès que des événements sont notifiés sur le réseau, il notifie alors les listeners ayant souscrit à ce type d’événements.

# References

[1] OSGi Alliance, OSGi Service Compendium, Release 4 version 4.3, May 2012[2] OSGi Alliance, RFC 192 ZigBee Device Service, ongoing specification. https://github.com/osgi/design/raw/master/rfcs/rfc0192/rfc-0192-ZigBeeDeviceService.pdf