Playzilla – Fonction boîtier 4 touches

Septembre 2011
Version 001A

SOMMAIRE

I. Boîtier 4 touches

Présentation du boîtier

Le boîtier de commande 4 touches est un boîtier qui se branche sur un port USB de la plateforme playzilla (elinux ou windows) :

Signification des touches

Ce boîtier est vu comme un clavier numérique usb, dont la correspondance entre les touches et le caractère est la suivante :

'/' pour le bouton rouge
'*' pour le bouton jaune
'-' pour le bouton vert
'+' pour le bouton bleu

Si on utilise un clavier usb classique à la place du boîtier 4 touches, l'appui sur l'un de ces 4 caractères aura le même effet.

II. Configuration de la fonction

Pré-requis

- L'utilisation de la fonction boîtier 4 touches nécessite une version de playzilla au moins égale à 2.50.76, et une configuration plugncast (serveur + studio).
- Avec playzilla, les événements claviers ne sont pas reçus lorsqu'un PowerPoint est actif. On ne peut donc pas utiliser le boîtier 4 touches pendant qu'un PowerPoint est joué.

1. Explication de la fonction

Cette fonction permet :

- Lors de l'appui sur une des 4 touches, le remplacement du contenu courant, visuel, ou audio, ou audio-visuel.
- Une nouvelle playlist constituée d'une scène est alors jouée à la place du contenu courant, jusqu'à sa fin. Une fois terminée, le contenu standard recommence à son début.
- Si pendant la playlist de remplacement on appuie à nouveau sur la même touche, la playlist de remplacement s'arrête avant sa fin, et le contenu standard recommence à son début.
- Si pendant la playlist de remplacement on appuie à nouveau sur une autre touche, la playlist de remplacement s'arrête avant sa fin, et la playlist de remplacement de la nouvelle touche se lance.

Il existe 3 types de playlist de remplacement :

- type 'audio' : ce type permet de remplacer le canal audio courant, sans affecter le canal audio-visuel. Si il n'y a pas de canal audio, ce type de playlist ne fonctionnera pas.
- type 'video' : ce type permet de remplacer le canal audio-visuel, sans affecter le canal audio. Si cette playlist est constituée d'une scène avec du son, ce son ne sera pas joué, seul le son du canal audio sera joué.
- type 'audio-video' : ce type permet de remplacer le canal audio-visuel et le canal audio. Si cette playlist est constituée d'une scène avec du son, ce son sera joué.

Remarques sur les touches :

- Le service utilisé cumule tous les caractères reçus pendant 0.5s. Il est donc possible d'ajouter des commandes activées par l'appui de plusieurs boutons successifs : par exemple l'appui du bouton rouge puis vert produit la séquence "/-".
- Le clavier 4 touches est géré comme un clavier standard et dispose donc de la répétition du code clavier si le bouton reste enfoncé. Pour générer la séquence de caractère "/", il faut donc un appui bref sur la touche rouge; si le bouton reste appuyé, d'autres caractères '/' seront ajoutés à la séquence et la commande prévue ne sera pas appelée.

2. Configuration dans le Studio

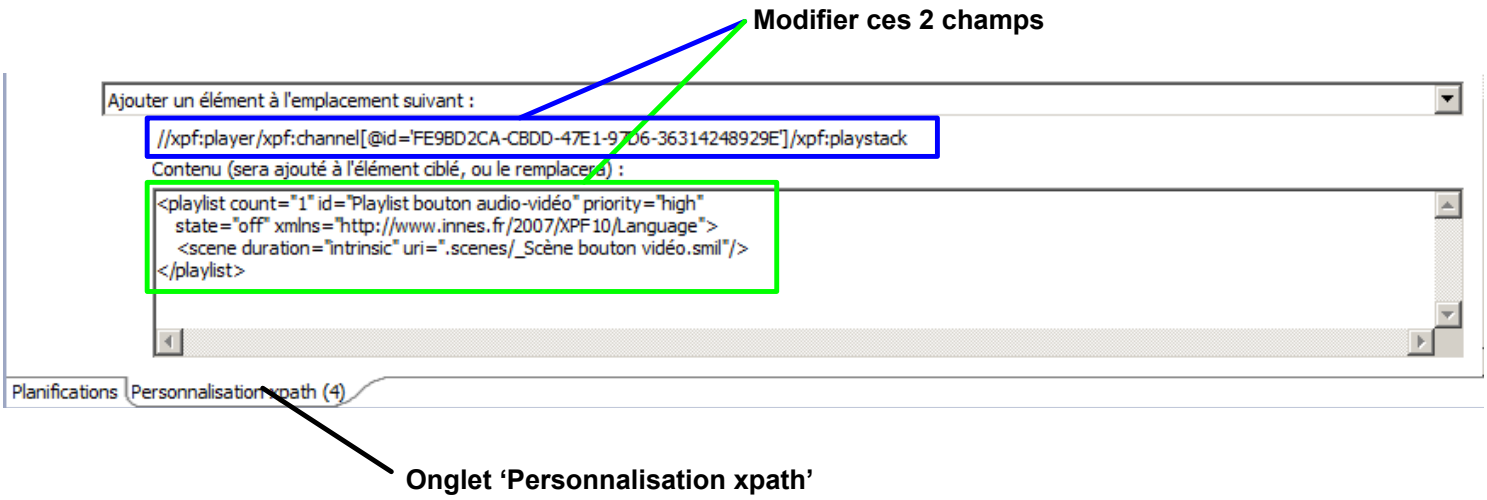
2.1. Point de départ de l'exemple

Pour illustrer la configuration à effectuer, nous partirons de l'exemple suivant :

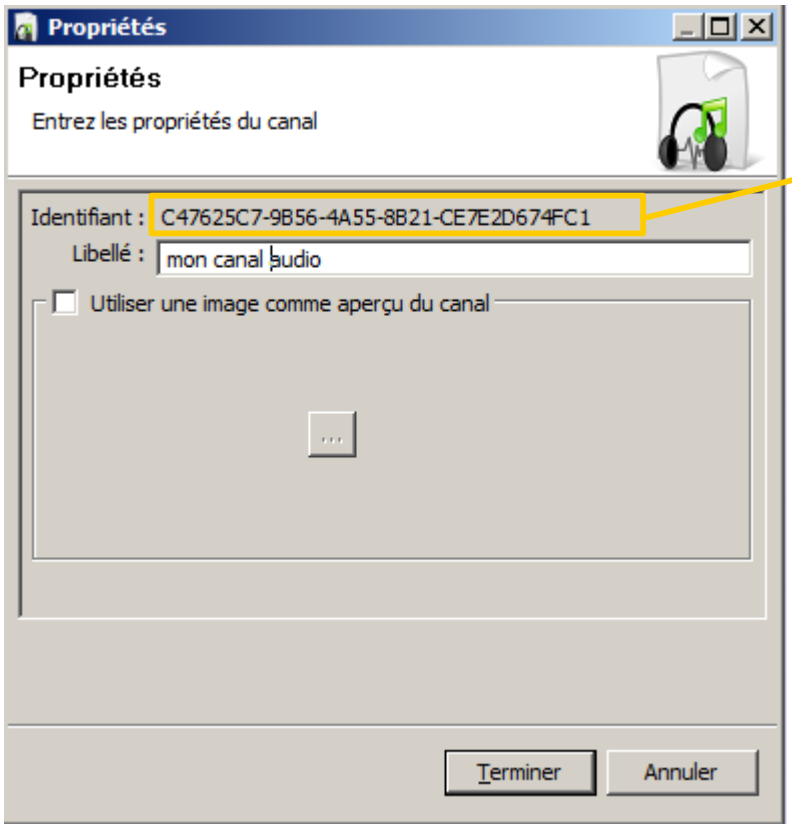
- Une programmation avec 2 canaux (1 canal audio-visuel, nommé 'mon canal audio-visuel' et 1 canal audio, nommé 'mon canal audio')
- Une scène audio '_Scène bouton audio' qui servira de contenu de remplacement
- Une scène audio-visuelle '_Scène bouton vidéo' qui servira de contenu de remplacement (scène constitué d'une vidéo avec du son)

2.2. Chargement des playlist de remplacement

Pour pouvoir charger dans la plateforme playzilla les scènes qui constituent les playlist de remplacement, il faut les déclarer explicitement.
Pour cela, dans le Studio, aller dans l'onglet 'Tâches', sélectionner la tâche de 'Mise à jour de la grille des programmes' correspondant au playzilla recevant les 2 canaux. Sur cette tâche, sélectionner l'onglet 'Personnalisation xpath'.
Cliquer sur l'icône 'Ajouter', et modifier les champs comme suit :



1ère partie (bleue) :
Elle sert à rajouter la partie verte dans l'élément 'playstack' du canal. Cette playstack doit être attachée à un canal existant. Chaque canal possède un identifiant, que l'on peut récupérer dans l'onglet 'Canaux', clic droit sur le canal concerné :



L'identifiant du canal est à insérer à la place du texte en rouge de cet exemple :
//xpf:player/xpf:channel[@id='FE9BD2CA-CBDD-47E1-97D6-36314248929E']/xpf:playstack

Comme il y a 2 canaux dans la 'mise à jour de la grille des programmes', l'association doit être effectuée comme ceci :

- playlist de remplacement de type 'audio' : l'attacher au canal audio
- playlist de remplacement de type 'video' : l'attacher au canal audio-visuel
- playlist de remplacement de type 'audio-video' : l'attacher au canal audio-visuel

Se référer à l'annexe pour avoir l'exemple complet.

2ème partie (verte) :
C'est la description de la playlist.
Exemple :

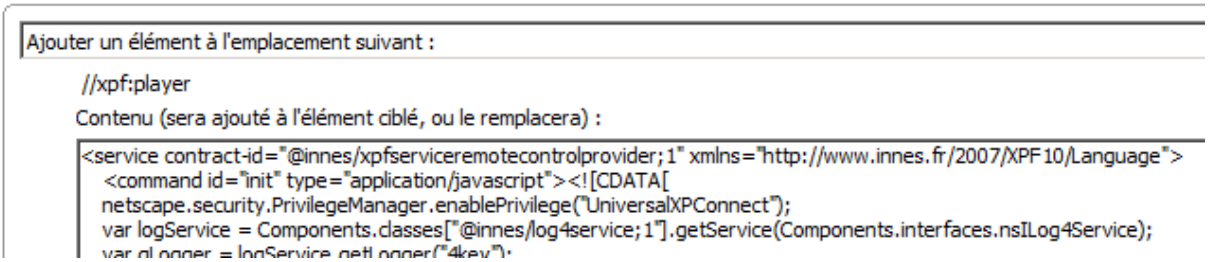
```
<playlist count="1" id="Playlist bouton audio-vidéo" priority="high"
   state="off" xmlns="http://www.innes.fr/2007/XPF10/Language">
   <scene duration="intrinsic" uri=".scenes/_Scène bouton vidéo.smil"/>
</playlist>
```

Explication des différents champs :

- count="1" : vaut "1" si l'on désire jouer 1 fois la playlist.
- id="Playlist bouton audio-vidéo" : c'est l'identifiant de la playlist, à remplacer par un identifiant libre, qui servira dans le paragraphe suivant.
- uri=".scenes/_Scène bouton vidéo.smil" : c'est le nom de la scène qui constitue cette playlist. Remplacer '_Scène bouton vidéo' par le nom de la scène de cette playlist.

2.3. Association des boutons avec les playlist de remplacement

Une fois les playlist de remplacement déclarées, il faut rajouter un dernier élément permettant d'activer le service de détection des touches, et d'associer chaque touche à la bonne playlist.
Ceci est effectué grâce à un nouvel élément dans 'Personnalisation xpath' :



Le contenu de cette partie est à prendre intégralement à partir de l'annexe de ce document, ou du fichier joint 'xpath_4key.xml'.
Seule la fin de ce contenu est à modifier :

```
]]></command>
<command id="/" type="application/javascript"><![CDATA[change_playlist("Playlist bouton audio", "audio");]]></command>
<command id="*" type="application/javascript"><![CDATA[change_playlist("Playlist bouton vidéo", "video");]]></command>
<command id="-" type="application/javascript"><![CDATA[change_playlist("Playlist bouton audio-vidéo", "audio-video");]]></command>
</service>
```

Dans cet exemple, il y a 3 touches associées à 3 playlist de remplacement :

- id="/" (bouton rouge) est associée à la playlist d'identifiant "Playlist bouton audio" qui est de type "audio".
- id="*" (bouton jaune) est associée à la playlist d'identifiant "Playlist bouton vidéo" qui est de type "video".
- id="-" (bouton vert) est associée à la playlist d'identifiant "Playlist bouton audio-vidéo" qui est de type "audio-video".

La 4ème touche n'aura donc aucun effet.

III. Annexe

1. Configuration du test

- Une programmation avec 2 canaux. Id du canal audio-visuel=’FE9BD2CA-CBDD-47E1-97D6-36314248929E’, id du canal audio=’C47625C7-9B56-4A55-8B21-CE7E2D674FC1’
- Le bouton rouge active la playlist ‘Playlist bouton audio’, de type ‘audio’, contituée de la scène ' _Scène bouton audio’
- Le bouton jaune active la playlist ‘Playlist bouton vidéo’, de type ‘video’, contituée de la scène ' _Scène bouton vidéo’
- Le bouton vert active la playlist ‘Playlist bouton audio-vidéo’, de type ‘audio-video’, contituée de la scène ' _Scène bouton vidéo’

2. Personnalisation xpath : chargement des playlists

Elle est contituée de 3 éléments :

- 1 -

//xpf:player/xpf:channel[@id='C47625C7-9B56-4A55-8B21-CE7E2D674FC1']/xpf:playstack

```
<playlist count="1" id="Playlist bouton audio" priority="high"
   state="off" xmlns="http://www.innes.fr/2007/XPF10/Language">
   <scene duration="intrinsic" uri=".scenes/_Scène bouton audio.smil"/>
</playlist>
```

- 2 -

//xpf:player/xpf:channel[@id='FE9BD2CA-CBDD-47E1-97D6-36314248929E']/xpf:playstack

```
<playlist count="1" id="Playlist bouton vidéo" priority="high"
   state="off" xmlns="http://www.innes.fr/2007/XPF10/Language">
   <scene duration="intrinsic" uri=".scenes/_Scène bouton vidéo.smil"/>
</playlist>
```

- 3 -

//xpf:player/xpf:channel[@id='FE9BD2CA-CBDD-47E1-97D6-36314248929E']/xpf:playstack

```
<playlist count="1" id="Playlist bouton audio-vidéo" priority="high"
   state="off" xmlns="http://www.innes.fr/2007/XPF10/Language">
   <scene duration="intrinsic" uri=".scenes/_Scène bouton vidéo.smil"/>
</playlist>
```

3. Personnalisation xpath : association boutons et playlists

Elle est constituée d’un élément :

//xpf:player

```
<service contract-id="@innes/xpfserviceremotecontrolprovider;1" xmlns="http://www.innes.fr/2007/XPF10/Language">
   <command id="init" type="application/javascript"><![CDATA[
        netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect");
        var logService = Components.classes["@innes/log4service;1"].getService(Components.interfaces.nsILog4Service);
        var gLogger = logService.getLogger("4key");
        var gPrefService = Components.classes["@mozilla.org/preferences-
service;1"].getService(Components.interfaces.nsIPrefBranch2);
        var xpfUtil = Components.classes["@innes/xpfutil;1"].getService(Components.interfaces.nsIXpfUtil);
        playlist_id="";
        playlist_type="";
        playlist=null;
        audio_channel=null;
        video_channel=null;
        inited = false;

        function ERROR(string)
        {
                netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect");
                gLogger.error (string, null);
        }
        function LOG(string)
        {
                netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect");
                gLogger.debug (string, null);
        }
        function listener(event)
        {
                LOG("handleEvent : " + event);
                if (event.type == "DOMAttrModified" &&
                        event.attrName == "state")
                        {
                                if (event.prevValue != event.newValue &&
                                        event.newValue == "off")
                                {
                                        LOG("listener: end of playlist");
                                        stop_playlist();
                                }
                        }
        }
        function change_playlist(name, type)
        {
                netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect");
                LOG("change_playlist : name = " + name + " type = " + type);
                if (!inited)
                {
                        ERROR("change_playlist : service not initialized");
                        return;
                }
                if (type != "audio" && type != "video" && type != "audio-video")
                {
                        ERROR("change_playlist : bad type of playlist : " + type);
                        return;
                }
                var t= (type == "audio") ? "audio" : "visual";
                try
                {
                        var str = "./xpf:playstack/xpf:playlist[@id='"+ name + "']";
                        str = "//xpf:player/xpf:channel[@type='" + t + "']/xpf:playstack/xpf:playlist[@id='"+ name +
"']";
                        var new_playlist=xpfUtil.getXpfNodeXPath(document, str, "xpf");
                        if (name == playlist_id)
                        {
                                LOG("change_playlist: stopping current playlist");
                                stop_playlist();
                                return;
                        }
                        else
```

```
                        stop_playlist();
                playlist = new_playlist;
                playlist_id = name;
                playlist_type = type;
                if (playlist_type == "audio-video" && audio_channel)
                {
                        var elt = audio_channel.QueryInterface(Components.interfaces.nsIDOMElement);
                        elt.setAttribute("state", "stop");
                }
                LOG("change_playlist : set state to on");
                var element = playlist.QueryInterface(Components.interfaces.nsIDOMElement);
                element.setAttribute("state", "on");
                var target = playlist.QueryInterface(Components.interfaces.nsIDOMEventTarget);
                target.addEventListener("DOMAttrModified", listener, false);
        }
        catch (ex)
        {
        if (new_playlist==null)
        {
                ERROR("Can't find playlist with id '" + name + "'");
        }
                ERROR("ex = " + ex);
        }
}
function stop_playlist()
{
        LOG("stop_playlist");
        if (playlist == null)
        {
                return;
        }
        if (playlist_type == "audio-video" && audio_channel)
        {
                var elt = audio_channel.QueryInterface(Components.interfaces.nsIDOMElement);
                elt.setAttribute("state", "play");
        }
        var target = playlist.QueryInterface(Components.interfaces.nsIDOMEventTarget);
        target.removeEventListener("DOMAttrModified", listener, false);
        var element = playlist.QueryInterface(Components.interfaces.nsIDOMElement);
        element.setAttribute("state", "off");
        playlist= null;
        playlist_type="";
        playlist_id="";
}

LOG("init");
try
{
video_channel= xpfUtil.getXpfNodeXPath(document, "//xpf:player/xpf:channel[@type='visual']", "xpf");
inited = true;
audio_channel= xpfUtil.getXpfNodeXPath(document, "//xpf:player/xpf:channel[@type='audio']", "xpf");
}
catch (ex)
{
if (inited)
        ERROR("Can't find audio channel");
else
        ERROR("Can't find video channel");
}

]]></command>
   <command id="/" type="application/javascript"><![CDATA[change_playlist("Playlist bouton audio",
"audio");]]></command>
   <command id="*" type="application/javascript"><![CDATA[change_playlist("Playlist bouton vidéo",
"video");]]></command>
   <command id="-" type="application/javascript"><![CDATA[change_playlist("Playlist bouton audio-vidéo", "audio-
video");]]></command>
</service>
```

Pour tout complément d'information, notre support technique se tient à votre disposition au ***02.23.20.01.62***

Contact support : support@innes.fr
Contact commercial : sales@innes.fr

INNES SAS
Immeuble Gallium
80 Avenue des Buttes de Coësmes
F-35700 RENNES

Tel : +33 (0)2 23 20 01 62
Fax : +33 (0)2 23 20 22 59

www.innes.fr