Intégrer une IPX800v3 sans passer par le plugin dans JEEDOM ATLAS

Message par **Seb82** » 21 août 2019, 23:03

Bonjour à tous,

Un peu échaudé par un nouveau problème sur le plugin de l'IPX800v3, je me suis décidé à voir comment l'intégrer (assez simplement) sans passer par ce dernier. Je partage ci-dessous ce que j'ai fait :

- Créer à la racine un objet nommé "IPX800" (Outils > Objets > Ajouter)
- Créer un virtuel nommé "IPX800_virtuel" (Plugins > Programmation > Virtuel > Ajouter) et choisir "Test" comme objet parent
- Dans ce virtuel, créer 4 commandes virtuelles nommées AN1 à AN4 avec comme sous-type "Numérique" => ce sont les entrées analogiques
- Dans ce virtuel, créer 4 commandes virtuelles nommées C1 à C4 avec comme sous-type "Numérique" => ce sont les compteurs
- Dans ce virtuel, créer 8 commandes virtuelles nommées R1 à R8 avec comme sous-type "Binaire" => ce sont l'état des relais
- Dans ce virtuel, créer 8 commandes virtuelles nommées I1 à I8 avec comme sous-type "Binaire" => ce sont les entrées digitales
- Créer un nouveau scénario (Outils > Scénarios > Ajouter)
- Choisir comme mode du scénario "Programmé" puis dans programmation mettre */5 * * * * (exécution toutes les 5 minutes, à vous de voir), et cocher "Multi-lancement"
- Dans l'onglet "Scénario", cliquer sur "Ajouter un bloc" et choisir "Code"
- Ajouter le code suivant en n'oubliant pas de modifer la partie configuration

```
/* IPX800v3

Ce code permet de mettre à jour un objet virtuel réprésentant l'IPX800 de deux manières :
- Soit par lecture du fichier globalstatus.xml sur l'IPX
- Soit par lancement par le push de l'IPX de ce scénario en transmettant un tag nommé push (permet d'avoir un retour immédiat)

Plus d'informations ici : https://www.jeedom.com/forum/viewtopic.php?f=133&t=47112

*/

/* Configuration */

$IP_IPX800 = "192.168.1.100";      // Changer par l'IP de l'IPX800
$PORT_IPX800 = "80";               // Port de l'IPX800 si modifié, sinon laisser 80
```

```
$IDENTIFIANT = "";                  // Identifiant s'il y a en a un, sinon vide
$MOT_DE_PASSE = "";                 // Mot de passe s'il y a en a un, sinon vide

/* Pour le push, dans la configuration de l'IPX aller dans M2M > Push, mettre l'IP
de Jeedom dans Server et dans Path mettre /core/api/jeeApi.php?
apikey=CLE_API&type=scenario&id=XX&action=start&tags=push%3De$O$I
 CLE_API est à remplacer par le clé API de Jeedom (roue crantée en haut à droite >
Configuration > API > Clef API )
 XX est à remplacer par l'ID de ce scénario
 $O contient l'état des 32 relais (8 de l'IPX800 et les éventuelles extensions)
 $I contient l'état des 32 entrées digitales (8 de l'IPX800 et les éventuelles
extensions)
*/

// On récupère les tags du scénario
$tags = $scenario->getTags();
// On extrait l'éventuel tag push
$push = $tags['#push#'];

// S'il n'y a pas de tag "push", on lit le fichier xml de l'IPX800

if (!$push) {

  $scenario->setLog("Pas de tag push, lecture du fichier globalstatus.xml sur
l'IPX800");

  // Url du fichier globalstatus.xml de l'IPX800
  $Url = "http://".(($IDENTIFIANT == "")?"":($IDENTIFIANT.":".$MOT_DE_PASSE."@")).
$IP_IPX800.":".$PORT_IPX800."/globalstatus.xml";
  $scenario->setLog("Url du fichier : ".$Url);

  // Récupération du fichier avec curl
  $InitPage = curl_init();
  curl_setopt($InitPage, CURLOPT_URL, $Url);
  curl_setopt($InitPage, CURLOPT_RETURNTRANSFER, 1);
  curl_setopt($InitPage, CURLOPT_USERAGENT, 'Jeedom');
  $resultat = curl_exec ($InitPage);
  //$scenario->setLog($resultat); // Décommenter cette ligne pour vérifier dans le
log du scénario le contenu récupéré
  $response_code = curl_getinfo($InitPage,CURLINFO_RESPONSE_CODE); // Récupération
du statut de réponse de la requête http
  curl_close($InitPage);

  if ($response_code == 200) { // Vérifie si la récupération s'est bien passée

    // Conversion du XML en objet PHP
    $xml = simplexml_load_string($resultat);

    // Extraction de la version pour voir si le fichier est valide
    $version = (string)$xml->version;
    $scenario->setLog("Version : ".$version);

      if ($version == "") {
      $scenario->setLog("Fichier globalstatus.xml non valide");
      log::add('ipx800','error',"Fichier globalstatus.xml non valide");
    } else {

      // Extraction des valeurs analogiques depuis XML
      $scenario->setLog("Mise à jour des valeurs analogiques depuis XML");
      cmd::byString("#[IPX800][IPX800_virtuel][AN1]#")->event((string)$xml-
>analog0);
      cmd::byString("#[IPX800][IPX800_virtuel][AN2]#")->event((string)$xml-
>analog1);
      cmd::byString("#[IPX800][IPX800_virtuel][AN3]#")->event((string)$xml-
>analog2);
      cmd::byString("#[IPX800][IPX800_virtuel][AN4]#")->event((string)$xml-
>analog3);
```

```
      // Extraction des compteurs depuis XML
      $scenario->setLog("Mise à jour des compteurs depuis XML");
      cmd::byString("#[IPX800][IPX800_virtuel][C1]#")->event((string)$xml->count0);
      cmd::byString("#[IPX800][IPX800_virtuel][C2]#")->event((string)$xml->count1);
      cmd::byString("#[IPX800][IPX800_virtuel][C3]#")->event((string)$xml->count2);
      cmd::byString("#[IPX800][IPX800_virtuel][C4]#")->event((string)$xml->count3);

      // Extraction des états des relais depuis XML
      $scenario->setLog("Mise à jour de l'état des relais depuis XML");
      cmd::byString("#[IPX800][IPX800_virtuel][R1]#")->event((string)$xml->led0);
      cmd::byString("#[IPX800][IPX800_virtuel][R2]#")->event((string)$xml->led1);
      cmd::byString("#[IPX800][IPX800_virtuel][R3]#")->event((string)$xml->led2);
      cmd::byString("#[IPX800][IPX800_virtuel][R4]#")->event((string)$xml->led3);
      cmd::byString("#[IPX800][IPX800_virtuel][R5]#")->event((string)$xml->led4);
      cmd::byString("#[IPX800][IPX800_virtuel][R6]#")->event((string)$xml->led5);
      cmd::byString("#[IPX800][IPX800_virtuel][R7]#")->event((string)$xml->led6);
      cmd::byString("#[IPX800][IPX800_virtuel][R8]#")->event((string)$xml->led7);
      cmd::byString("#[IPX800][IPX800_virtuel][R9]#")->event((string)$xml->led8);
      cmd::byString("#[IPX800][IPX800_virtuel][R10]#")->event((string)$xml->led9);
      cmd::byString("#[IPX800][IPX800_virtuel][R11]#")->event((string)$xml->led10);
      cmd::byString("#[IPX800][IPX800_virtuel][R12]#")->event((string)$xml->led11);
      cmd::byString("#[IPX800][IPX800_virtuel][R13]#")->event((string)$xml->led12);
      cmd::byString("#[IPX800][IPX800_virtuel][R14]#")->event((string)$xml->led13);
      cmd::byString("#[IPX800][IPX800_virtuel][R15]#")->event((string)$xml->led14);
      cmd::byString("#[IPX800][IPX800_virtuel][R16]#")->event((string)$xml->led15);

      // Extraction des entrées digitales depuis XML
      $scenario->setLog("Mise à jour des entrées digitales depuis XML");
      cmd::byString("#[IPX800][IPX800_virtuel][I1]#")->event(((string)$xml-
>btn0=="up")?0:1);
      cmd::byString("#[IPX800][IPX800_virtuel][I2]#")->event(((string)$xml-
>btn1=="up")?0:1);
      cmd::byString("#[IPX800][IPX800_virtuel][I3]#")->event(((string)$xml-
>btn2=="up")?0:1);
      cmd::byString("#[IPX800][IPX800_virtuel][I4]#")->event(((string)$xml-
>btn3=="up")?0:1);
      cmd::byString("#[IPX800][IPX800_virtuel][I5]#")->event(((string)$xml-
>btn4=="up")?0:1);
      cmd::byString("#[IPX800][IPX800_virtuel][I6]#")->event(((string)$xml-
>btn5=="up")?0:1);
      cmd::byString("#[IPX800][IPX800_virtuel][I7]#")->event(((string)$xml-
>btn6=="up")?0:1);
      cmd::byString("#[IPX800][IPX800_virtuel][I8]#")->event(((string)$xml-
>btn7=="up")?0:1);
    }

  } else {
    $scenario->setLog("Erreur lors de la récupération du fichier
globalstatus.xml");
    log::add('ipx800','error',"Erreur lors de la récupération du fichier
globalstatus.xml");
  }

} else {

  $scenario->setLog("Tag push détecté, information reçue : ".$push);

  if (substr($push,0,1) == 'e') {  // On vérifie que le tag push commence par un
'e' pour être sûr d'avoir une bonne information

    // Extraction des états des relais depuis tag push -
    $scenario->setLog("Mise à jour de l'état des relais depuis tag");
    cmd::byString("#[IPX800][IPX800_virtuel][R1]#")->event(substr($push,1,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R2]#")->event(substr($push,2,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R3]#")->event(substr($push,3,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R4]#")->event(substr($push,4,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R5]#")->event(substr($push,5,1));
```

```
    cmd::byString("#[IPX800][IPX800_virtuel][R6]#")->event(substr($push,6,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R7]#")->event(substr($push,7,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R8]#")->event(substr($push,8,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R9]#")->event(substr($push,9,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R10]#")->event(substr($push,10,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R11]#")->event(substr($push,11,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R12]#")->event(substr($push,12,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R13]#")->event(substr($push,13,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R14]#")->event(substr($push,14,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R15]#")->event(substr($push,15,1));
    cmd::byString("#[IPX800][IPX800_virtuel][R16]#")->event(substr($push,16,1));

    // Extraction des entrées digitales depuis tag push
    $scenario->setLog("Mise à jour des entrées digitales depuis tag");
    cmd::byString("#[IPX800][IPX800_virtuel][I1]#")->event(substr($push,32+1,1));
    cmd::byString("#[IPX800][IPX800_virtuel][I2]#")->event(substr($push,32+2,1));
    cmd::byString("#[IPX800][IPX800_virtuel][I3]#")->event(substr($push,32+3,1));
    cmd::byString("#[IPX800][IPX800_virtuel][I4]#")->event(substr($push,32+4,1));
    cmd::byString("#[IPX800][IPX800_virtuel][I5]#")->event(substr($push,32+5,1));
    cmd::byString("#[IPX800][IPX800_virtuel][I6]#")->event(substr($push,32+6,1));
    cmd::byString("#[IPX800][IPX800_virtuel][I7]#")->event(substr($push,32+7,1));
    cmd::byString("#[IPX800][IPX800_virtuel][I8]#")->event(substr($push,32+8,1));

  }

}
```

- Sauver et tester en cliquant sur Exécuter. Si ce scénario est exécuté sans tags (lancement normal), il récupère toutes les 5 minutes le fichier globalstatus.xml sur l'IPX800 et injecte l'information dans le virtuel IPX800_virtuel. Vérifier à ce stade que les valeurs sont bien récupérées.
- Maintenant, on va configurer le retour d'état immédiat (push). Dans la configuration de l'IPX800, cliquer en bas sur M2M puis en bas encore sur Push, renseigner dans Server l'IP de Jeedom, cocher Enable, et dans Path entrer (en remplaçant CLE_API par votre Clef API - voir dans Jeedom roue crantée en haut à droite > Configuration > API > Clef API - et en remplaçant aussi XX par l'ID du scénario créé précédemment) :

Code : Tout sélectionner

```
/core/api/jeeApi.php?apikey=CLE_API&type=scenario&id=XX&action=start&tags=push%3De$O$I
```

- Cette configuration de push fait qu'à chaque fois qu'il y a une modification, l'IPX va lancer le scénario précédemment créé. En plus, elle va transmettre l'état des sorties ($O pour outputs) et des entrées digitales ($I pour inputs) à l'aide d'un tag nommé "push". Dans ce cas, le scenario va extraire l'information transmise et mettre à jour le virtuel IPX800_virtuel.

Cette configuration ne prend toutefois pas en compte d'éventuelles extensions de l'IPX800, mais le principe est le même, il suffit d'adapter le code (toute l'information est disponible dans globalstatus.xml et dans le push). Bien sûr, je recommande de tout tester une fois mis en place.

Fait intéressant, on peut ensuite changer les noms de l'objet virtuel et de ses commandes : les changements sont reportés automatiquement dans le code du scénario. Pratique.

Autre remarque, pour les valeurs analogiques c'est la valeur brute (de 0 à 1023) qui est récupérée. Pour retrouver quelque chose de lisible, on peut par exemple dans la configuration de la commande, onglet Configuration, entrer une formule de calcul. Par exemple mettre "#value# / 1023 * 3.3" si on veut afficher la tension. Là c'est sûr que c'est un peu moins bien que ce que permettait le plugin puisqu'on pouvait directement sélectionner un type de sonde et que la formule était injectée dans la commande Réel => éventuellement il faudrait lister les différentes formules à appliquer. edit : voir 3 posts plus bas pour les formules

Il nous reste à créer les commandes pour commander les relais. Pour cela :
- Créer un nouveau script : Plugins > Programmation > Script > Ajouter, qu'on appellera par exemple IPX800_commandes
- Cliquer sur "Ajouter commande script", la nommer R1_ON, choisir HTTP comme Type script, choisir Type Action et laisser Défaut
- Dans Requête, entrer "http://192.168.1.100:80/preset.htm?set1=1" en remplaçant 192.168.1.100 par l'IP de votre IPX800
- Entrer éventuellement utilisateur et mot de passe si vous en avez défini un pour accéder à l'IPX800
- Créer une nouvelle commande script nommée R1_OFF avec comme requête cette fois-ci : http://192.168.1.100:80/preset.htm?set1=0
- Créer les commandes suivantes jusqu'à R8_ON et R8_OFF (vous pouvez utiliser l'icône dupliquer à droites des commandes précédentes pour aller plus vite) en changeant juste le numéro après set dans la requête. Par exemple, pour R8_OFF, ce sera http://192.168.1.100:80/preset.htm?set8=0

Et voilà. Vous devriez avoir deux objets qui permettent d'interagir avec votre IPX800v3 et de récupérer toutes les informations. Il ne reste plus qu'à remplacer les anciennes commandes, par exemple en utilisant "Cette commande remplace la commande" dans Configuration commande. C'est peut-être la partie la plus fastidieuse.

Normalement à partir de là, vous ne devriez plus avoir besoin du plugin et comme on utilise des composants intégrés au Core de Jeedom plus de problème avec les mises à jour.