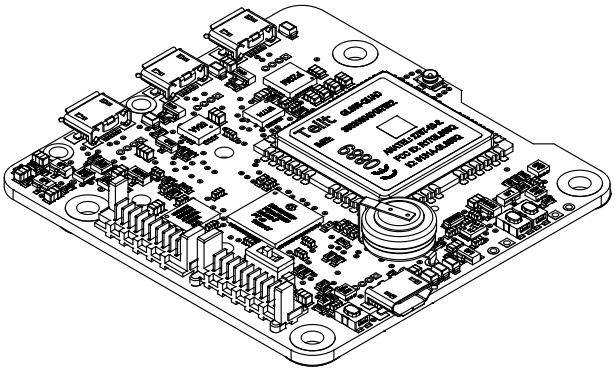
YoctoHub-GSM-3G-NA

Mode d'emploi

Table des matières

1. Introduction

Le YoctoHub-GSM-3G-NA est un module électronique de 60x58mm qui permet de contrôler d'autres modules Yoctopuce à travers une connection cellulaire de type GSM 3G (UMTS et HSPA). Le module radio supporte les deux bandes de fréquences GSM 850Mhz et 1900Mhz utilisés en Amérique du Nord, dans les Caraïbes et l'Amérique Latine[1]



Le YoctoHub-GSM-3G-NA

Le YoctoHub-GSM-3G-NA a été conçu pour être déployé facilement et ne pas demander de maintenance particulière. Contrairement à un mini-PC, il n'utilise pas un système d'exploitation complexe. Les réglages peuvent être effectués manuellement ou de manière automatisée, par USB. Il convient de ce fait beaucoup mieux à une industrialisation qu'un mini-PC. En revanche, il ne permet pas l'exécution de programmes supplémentaires écrits par l'utilisateur.

Le YoctoHub-GSM-3G-NA n'est pas un hub USB standard avec accès réseau. Bien qu'utilisant du câblage USB, ses ports descendants utilisent un protocole propriétaire, plus simple qu'USB. Il n'est par conséquent pas possible de contrôler, ni même d'alimenter, des périphériques USB standards avec un YoctoHub-GSM-3G-NA.

Yoctopuce vous remercie d'avoir fait l'acquisition de ce YoctoHub-GSM-3G-NA et espère sincèrement qu'il vous donnera entière satisfaction. Les ingénieurs Yoctopuce se sont donnés beaucoup de mal pour que votre YoctoHub-GSM-3G-NA soit facile à installer n'importe où et soit facile utiliser en toutes circonstances. Néanmoins, si ce module venait à vous décevoir, n'hésitez pas à contacter le support Yoctopuce[2].

[1] Pour une liste détaillée des bandes de fréquence supportées par pays, consultez la page Wikipedia http://en.wikipedia.org/wiki/GSM_frequency_bands .
[2] support@yoctopuce.com

1.1. Accessoires optionnels

Il existe un certain nombre d'accessoires qui vous aideront à tirer le meilleur parti de votre YoctoHub-GSM-3G-NA.

Vis et entretoises

Pour fixer le module YoctoHub-GSM-3G-NA à un support, vous pouvez placer des petites vis de 3mm avec une tête de 8mm au maximum dans les trous prévus ad-hoc. Il est conseillé de les visser dans des entretoises filetées, que vous pourrez fixer sur le support. Vous trouverez plus de détail à ce sujet dans le chapitre concernant le montage et la connectique. Micro-hub USB

Câble USB MicroB-MicroB

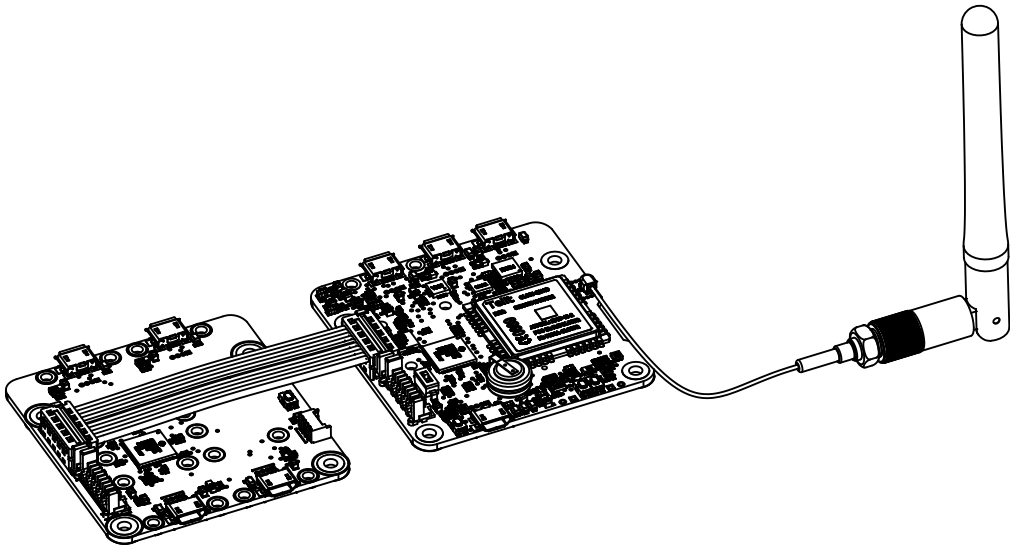
Les ports descendant du YoctoHub-GSM-3G-NA sont au format micro-B, tous les produits Yoctopuce sont équipés de connecteur micro-B. Cela signifie que vous aurez besoin d'un câble se terminaux par une prise Micro-B à chaque extrémité. Ces câbles ne sont pas très répandus dans le commerce, mais vous pourrez en trouver sur le magasin en ligne de Yoctopuce[3].

Connecteur au pas 1.27mm

L'utilisation de câbles USB pour connecter les sous modules est très pratique, mais cela reste une solution assez volumineuse. Sur le PCB du YoctoHub-GSM-3G-NA, vous trouverez a proximité de chaque connecteur USB une empreinte permettant de souder un connecteur au pas 1.27 ou 1.25mm. Ce qui permet d'interconnecter les modules avec un câblage beaucoup compact. Ce genre de connecteur est très standard, et peut être acheté dans à peu près n'importe que magasin d'électronique. Yoctopuce commercialise une ensemble connecteur + câble de 11cm sous la référence *1.27-1.27-11*.

YoctoHub-Shield

Le YoctoHub-GSM-3G-NA dispose de trois ports permettant de brancher trois sous-modules. Il est possible d'augmenter significativement cette capacité en utiliser des extension nommées *YoctoHub-Shield*. Chacun de ces shields ajoute quatre ports supplémentaires, et il est possible d'en chainer jusqu'à dix. Consulter la documentation du YoctoHub-Shield pour plus de détails.



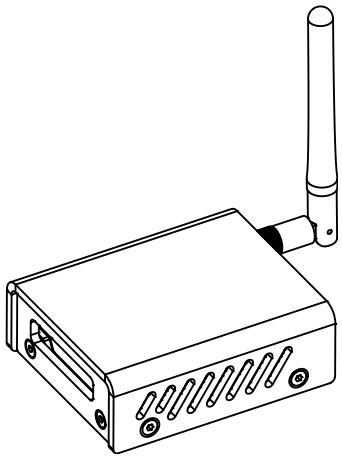
Le YoctoHub-Shield ajoute des ports à votre YoctoHub-GSM-3G-NA.

3

[3] *USB-OTG-MicroB-MicroB-20* et *USB-OTG-MicroB-MicroB-100*

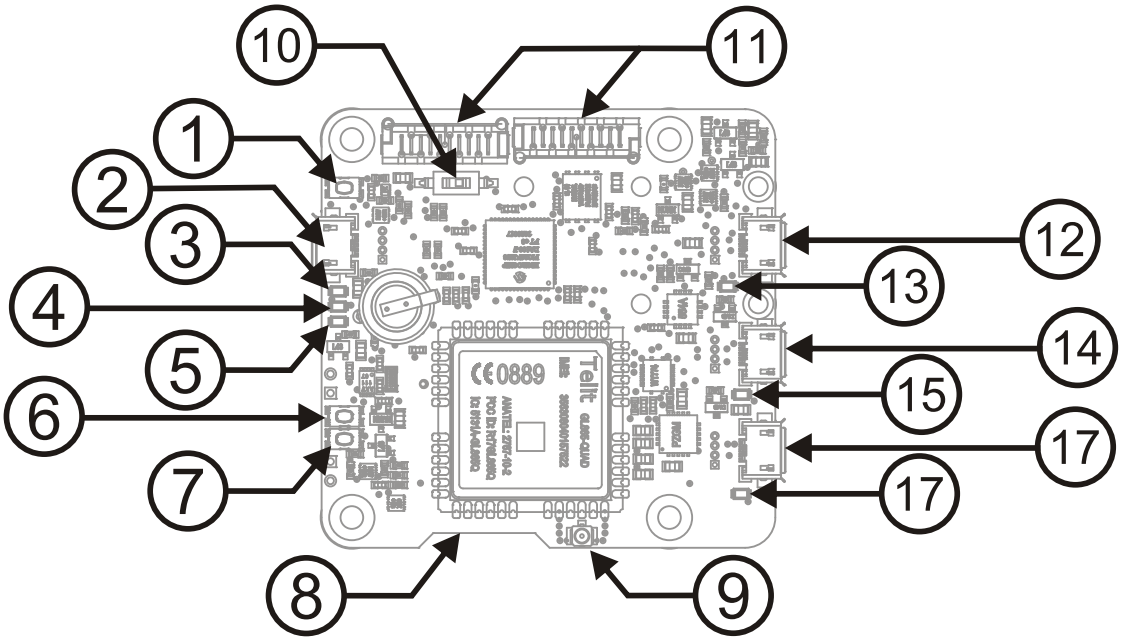
Boîtier

Votre YoctoHub-GSM-3G-NA a été conçu pour pouvoir être installé tel quel dans votre projet. Néanmoins Yoctopuce commercialise des boîtiers spécialement conçus pour les modules Yoctopuce. Vous trouverez plus d'informations à propos de ces boîtiers sur le site de Yoctopuce. Le boîtier recommandé pour votre YoctoHub-GSM-3G-NA est le modèle YoctoBox-HubWlan-Transp.



Votre YoctoHub-GSM-3G-NA peut être installé dans un boitier.

2. Présentation



1: Yocto-bouton
2: Port USB de contrôle + alimentation
3: Yocto-Led
4: Indicateur de sur-consommation
5: Indicateur de transfert réseau
6: Touche réveil
7: Touche mise en sommeil
8: Support pour carte SIM (dessous)
9: Connecteur d'antenne
10: Neutralisation de la mise en sommeil
11: Connexion dorsale
12: Port descendant 1
13: Indicateur port descendant 1
14: Port descendant 2
15: Indicateur port descendant 2
16: Port descendant 3
17: Indicateur port descendant 3

2.1. Les éléments du YoctoHub-GSM-3G-NA

Le numéro de série

Chaque Yocto-module a un numéro de série unique attribué en usine, pour les modules YoctoHub-GSM-3G-NA ce numéro commence par YHUBGSM4. Le module peut être piloté par logiciel en utilisant ce numéro de série. Ce numéro de série ne peut pas être changé.

Le nom logique

Le nom logique est similaire au numéro de série, c'est une chaîne de caractères sensée être unique qui permet référencer le module par logiciel. Cependant, contrairement au numéro de série, le nom logique peut être modifié à volonté. L'intérêt est de pouvoir fabriquer plusieurs exemplaires du même projet sans avoir à modifier le logiciel de pilotage. Il suffit de programmer les mêmes noms logiques dans chaque exemplaire. Attention, le comportement d'un projet devient imprévisible s'il contient plusieurs modules avec le même nom logique et que le logiciel de pilotage essaye d'accéder à l'un de ces modules à l'aide de son nom logique. A leur sortie d'usine, les modules n'ont pas de nom logique assigné, c'est à vous de le définir.

Le Yocto-bouton

Le Yocto-bouton a deux fonctions. Premièrement, il permet d'activer la Yocto-balise (voir la Yocto-Led ci-dessous). Deuxièmement, si vous branchez un Yocto-module en maintenant ce bouton appuyé, il vous sera possible de reprogrammer son firmware avec une nouvelle version. Notez qu'il existe une méthode plus simple pour mettre à jour le firmware depuis l'interface utilisateur, mais cette méthode-là peut fonctionner même lorsque le firmware chargé sur le module est incomplet ou corrompu.

La Yocto-Led

En temps normal, la Yocto-Led sert à indiquer le bon fonctionnement du module: elle émet alors une faible lumière bleue qui varie lentement mimant ainsi une respiration. La Yocto-Led cesse de respirer lorsque le module ne communique plus, par exemple s'il est alimenté par un hub sans connexion avec un ordinateur allumé.

Lorsque vous appuyez sur le Yocto-bouton, la Led passe en mode Yocto-balise: elle se met alors à flasher plus vite et beaucoup plus fort, dans le but de permettre une localisation facile d'un module lorsqu'on en a plusieurs identiques. Il est en effet possible de déclencher la Yocto-balise par logiciel, tout comme il est possible de détecter par logiciel une Yocto-balise allumée.

La Yocto-Led a une troisième fonctionnalité moins plaisante: lorsque le logiciel interne qui contrôle le module rencontre une erreur fatale, elle se met à flasher SOS en morse[1]. Dans ce cas, débranchez puis re-branchez le module. Si le problème venait à se reproduire, vérifiez que le module contient bien la dernière version du firmware et, dans l'affirmative, contactez le support Yoctopuce[2].

Le connecteur de contrôle et d'alimentation (Power / Control port)

Ce connecteur permet d'alimenter le YoctoHub-GSM-3G-NA et les modules qui lui sont connectés à l'aide d'un simple chargeur USB. Ce connecteur permet aussi de prendre le contrôle du YoctoHub-GSM-3G-NA par USB, exactement comme on pourrait le faire avec un module Yoctopuce classique. C'est particulièrement utile lorsque que l'on désire configurer le YoctoHub-GSM-3G-NA sans connaître son adresse IP.

Les ports descendants

Vous pouvez connecter jusqu'à trois modules Yoctopuce sur ces ports. Ils seront alors accessibles comme s'ils étaient branchés à un ordinateur faisant tourner un VirtualHub. Attention, le protocole entre le YoctoHub-GSM-3G-NA et le module Yoctopuce n'est pas de l'USB mais un protocole propriétaire plus léger. De ce fait le YoctoHub-GSM-3G-NA ne peut pas gérer des périphériques autres que des modules Yoctopuce. Un hub USB standard ne fonctionnera pas non plus[3]. Si vous désirez brancher plus de trois modules Yoctopuce, utilisez le connecteur dorsal pour y connecter un ou plusieurs YoctoHub-Shield[4].

Attention, les connecteurs USB du YoctoHub-GSM-3G-NA sont simplement soudés en surface et peuvent être arrachés si la prise USB venait à faire fortement levier. Si les pistes sont restées en place, le connecteur peut être ressoudé à l'aide d'un bon fer et de flux. Alternativement, vous pouvez

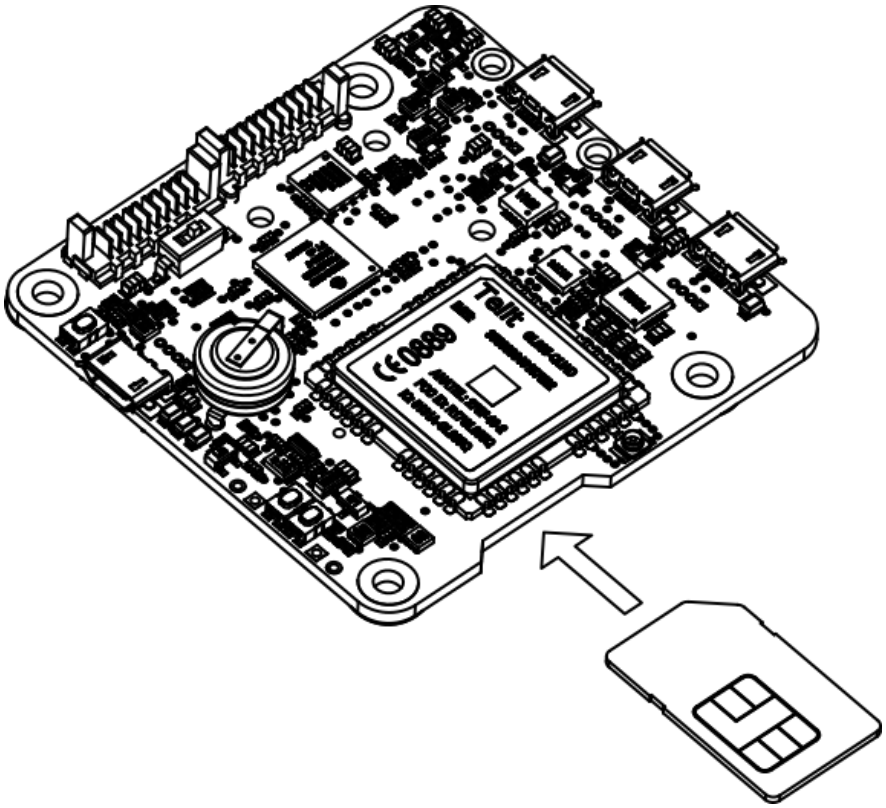
[1] court-court-court long-long-long court-court-court
[2] support@yoctopuce.com
[3] Le Micro-USB-Hub fabriqué par Yoctopuce est un hub USB standard et ne fonctionnera pas avec le YoctoHub-GSM-3G-NA.
[4] www.yoctopuce.com/FR/products/yoctohub-shield

souder un fil USB directement dans les trous espacés de 1.27mm prévus à cet effet, près du connecteur.

Le support pour la carte SIM

Pour vous connecter à un réseau cellulaire GSM, vous devrez insérer dans votre YoctoHub-GSM-3G-NA une carte SIM autorisant la connexion au réseau cellulaire, associée à un abonnement permettant le transfert de données. Le support à carte SIM est prévu pour le format mini-SIM le plus standard,aussi appelé 2FF. Il existe des adaptateurs permettant l'utilisation des Micro-SIM ou Nano-SIM, ces adaptateur peuvent être achetés dans n'importe quel magasin de téléphone portable. avec le YoctoHub-GSM-3G-NA. Le support SIM est de type *push-push*: pressez pour insérer la SIM jusqu'à-ce qu'elle soit en position et produise un petit clic. Re-pressez une deuxième fois éjecter la SIM de son support.

Vous devez insérer la carte SIM avec les contacts métalliques contre le circuit imprimé.

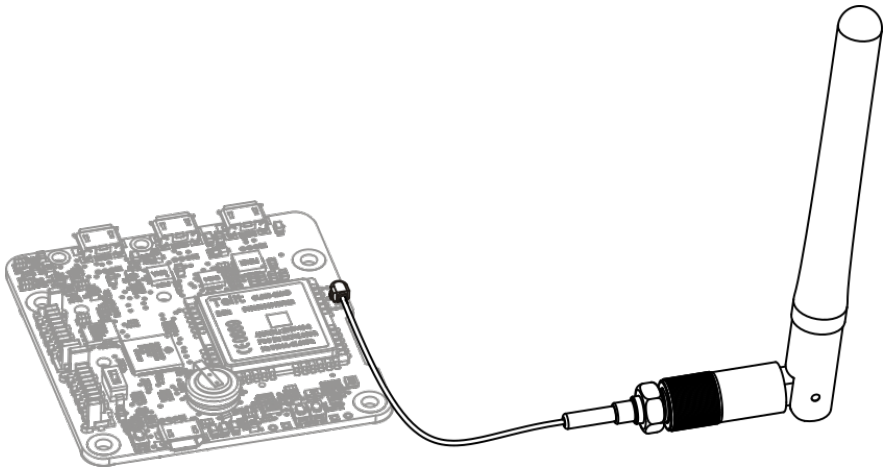


Sens d'insertion de la carte SIM dans le YoctoHub-GSM-3G-NA.

Le connecteur d'antenne

Le YoctoHub-GSM-3G-NA dispose d'un connecteur d'antenne coaxial ultra miniature (UFL). Prenez grand soin du connecteur UFL, il est fragile et n'est pas conçu pour supporter beaucoup de cycles de connexion/déconnexion. Le YoctoHub-GSM-3G-NA est livré en standard avec un petit câble UFL vers SMA femelle[5] et une antenne correspondante SMA mâle[6]. Vous pouvez utiliser une autre antenne de votre choix, pour autant qu'elle soit conçue pour la gamme de fréquence utilisée dans votre pays pour le GSM et qu'elle ait le bon connecteur. Prenez garde aussi au fait que l'utilisation d'antennes à fort gain peut vous amener à émettre un signal supérieur à la norme autorisée dans votre pays.

[5] filetage extérieur et tube femelle au centre
[6] filetage intérieur et pin mâle au centre



Connexion de l'antenne.

Indicateur de sur-consommation

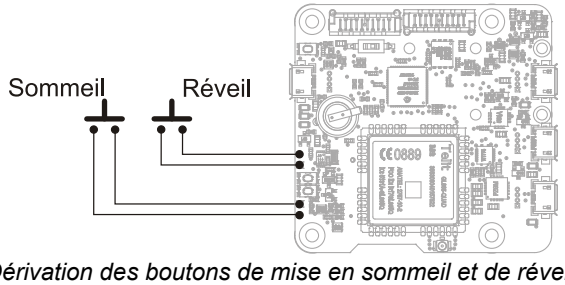
Le YoctoHub-GSM-3G-NA analyse en permanence sa consommation. S'il détecte une consommation globale de plus de 2A suite à une surcharge sur un des ports descendants par exemple, il va automatiquement désactiver tous les ports descendants et allumer l'indicateur de sur-consommation. Pour isoler la source du problème, vous pouvez réactiver les ports un à un, en surveillant l'augmentation de la consommation. Alternativement, si connaissez la source du problème de sur-consommation et savez l'avoir résolu, vous pouvez redémarrer le YoctoHub-GSM-3G-NA pour réactiver tous les ports.

Notez que l'indicateur de sur-consommation est une mesure de protection qui peut éviter la surchauffe, mais ce n'est pas une garantie de protection contre les court- circuits.

Mise en sommeil

En moyenne, le YoctoHub-GSM-3G-NA consomme environ 0,5 Watt (100mA), auquel il faut ajouter la consommation des modules qui lui sont connectés. Mais il est capable de se mettre en sommeil pour réduire sa consommation d'énergie au strict minimum, et de se réveiller à une heure précise ou lorsqu'un contact extérieur est fermé. Cette fonctionnalité est très utile pour construire des installations de mesure fonctionnant sur batterie. Lorsque que le YoctoHub-GSM-3G-NA est en sommeil, la quasi totalité de l'électronique du module ainsi que les modules Yoctopuce connectés sont hors tension, ce qui réduit sa consommation totale à 75 µW (15 µA).

La mise en sommeil et le réveil peuvent être soit programmés sur base horaire, soit contrôlés par logiciel, soit contrôlés manuellement à l'aide de deux boutons poussoirs présents sur le circuit du YoctoHub-GSM-3G-NA. Vous y trouverez aussi deux paires de contacts qui permettent de dériver ces deux boutons.



Dérivation des boutons de mise en sommeil et de réveil.

Le YoctoHub-GSM-3G-NA dispose d'un interrupteur qui permet de désactiver au niveau hardware la fonctionnalité de mise en sommeil. Cette fonctionnalité est utile en particulier durant les phases de développement/déverminage de votre projet, ainsi que pour effectuer les mises à jour du firmware.

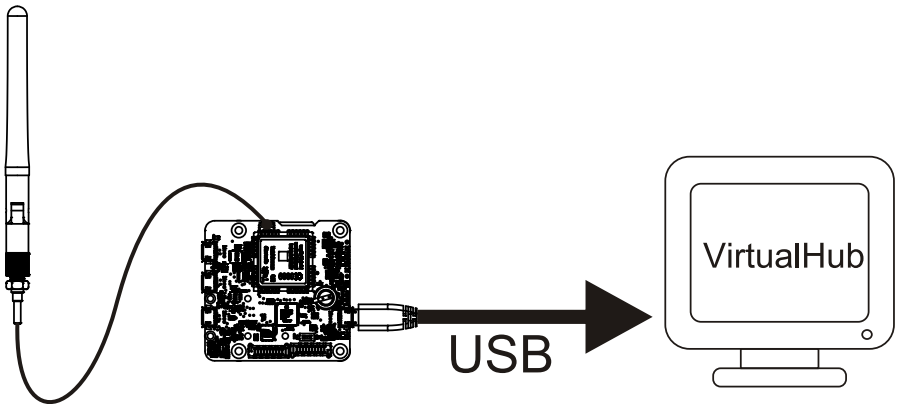
3. Premiers pas

Ce chapitre a pour but de vous aider à connecter et configurer votre YoctoHub-GSM-3G-NA pour la première fois

3.1. Configuration manuelle

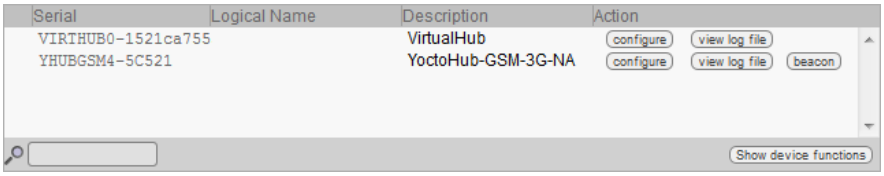
Vous pouvez configurer votre YoctoHub-GSM-3G-NA via son port de contrôle USB, en utilisant le *VirtualHub*[1].

Lancez un *VirtualHub* sur votre ordinateur favori et raccordez votre ordinateur au port *power / control port* du YoctoHub-GSM-3G-NA. Vous aurez besoin d'un câble USB A-MicroB.



Configuration: raccordez par USB votre YoctoHub-GSM-3G-NA à un ordinateur

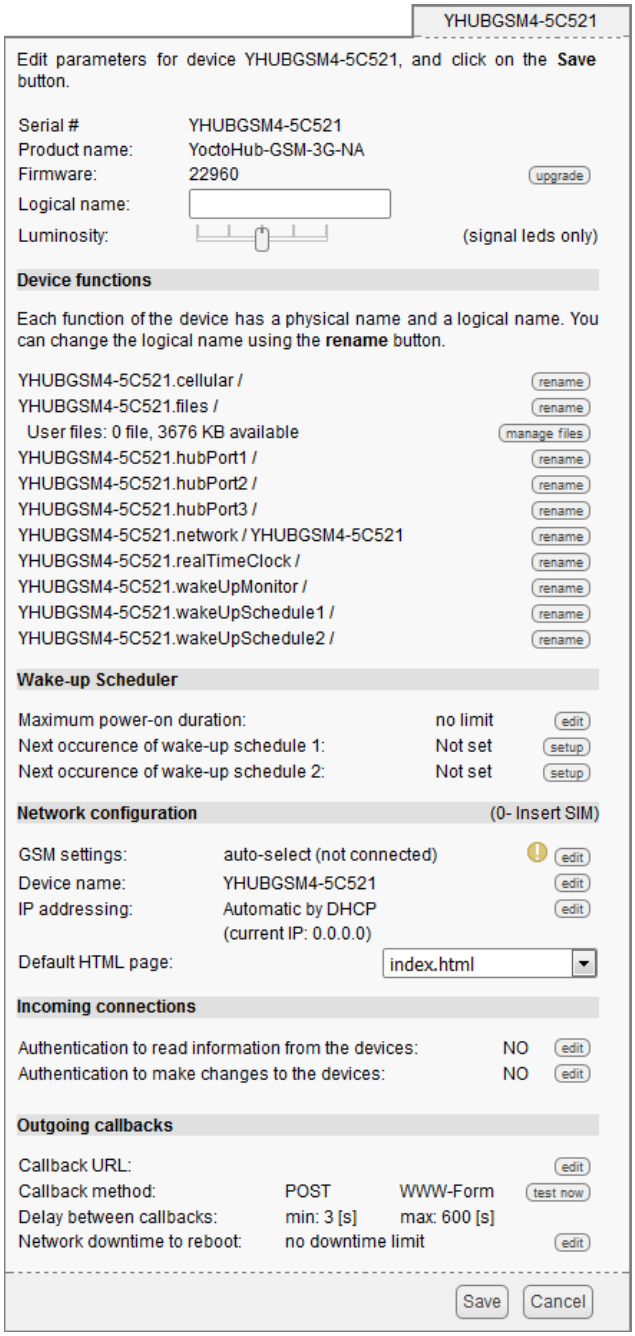
Lancez alors votre browser favori sur l'url de votre *VirtualHub*. Il s'agit généralement *http://127.0.0.1:4444*. Vous obtiendrez la liste des modules Yoctopuce connectés par USB, dont votre YoctoHub-GSM-3G-NA



Liste des modules Yoctopuce raccordés par USB à votre ordinateur, dont votre YoctoHub-GSM-3G-NA

[1] *http://www.yoctopuce.com/FR/virtualhub.php*

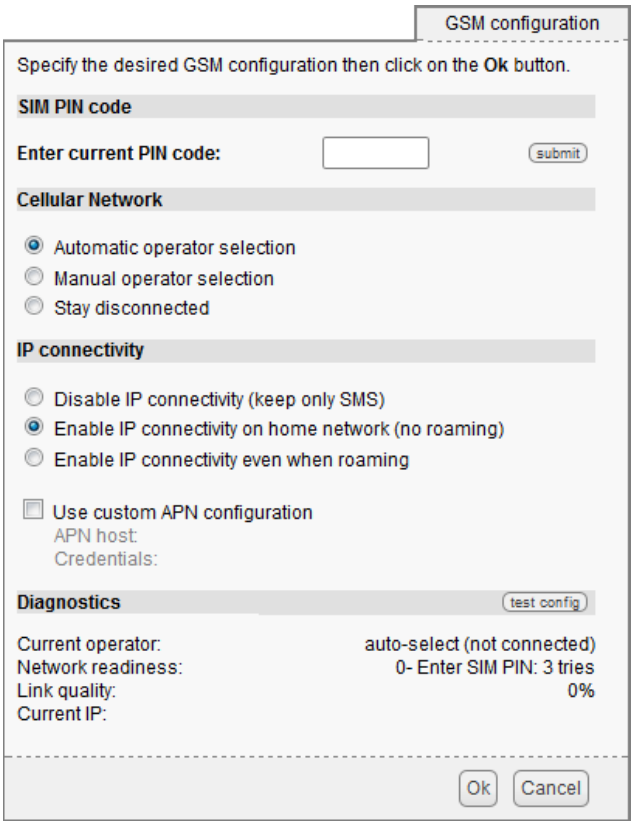
Cliquez sur le bouton **configure** correspondant à votre YoctoHub-GSM-3G-NA, votre obtiendrez la fenêtre de configuration du module. Cette fenêtre comporte une section **Network configuration**.



Fenêtre de configuration du module YoctoHub-GSM-3G-NA

Connexion au réseau cellulaire GSM

La première chose à faire consiste à configurer votre YoctoHub-GSM-3G-NA pour qu'il se connecte à votre réseau GSM. Pour cela cliquez sur le bouton **edit** correspondant à **GSM configuration** dans la section **Network configuration**, et la fenêtre de configuration du réseau cellulaire GSM apparaît:



Fenêtre de configuration du réseau cellulaire GSM.

Vous pouvez alors entrer le code PIN correspondant à la carte SIM introduite dans le YoctoHub-GSM-3G-NA si nécessaire, et choisir avec quel opérateur cellulaire vous désirez travailler.

Dans la plupart des cas, la cartes SIM "sait" pour quel opérateur elle est prévue, et il est possible de garder simplement la sélection automatique. Toutefois, en zone frontalière, il se peut que la carte soit tentée d'utiliser un émetteur étranger plus puissant, mais aussi plus cher à l'utilisation. Pour éviter cela, vous pouvez effectuer un choix d'opérateur manuel de votre réseau domestique.

Vous pouvez aussi spécifier dans votre YoctoHub-GSM-3G-NA le contexte quel cas vous désirez activer la connection IP (transfert de données). Vous pouvez soit la désactiver complètement si vous n'êtes intéressé qu'à l'utilisation de SMS[2], soit l'activer sur le réseau propre de la carte SIM, soit autoriser l'utilisation des données sur les réseaux tiers (roaming). Prenez garde si vous activez cette dernière option, car elle peut engendrer rapidement des coûts importants !

Selon votre SIM et votre opérateur cellulaire, il est possible que vous deviez configurer un APN (*Access Point Name*), qui correspond à la passerelle vers internet sur votre réseau cellulaire. Il s'agit d'un nom arbitraire, décidé par votre opérateur, et auquel s'ajoute parfois un nom d'utilisateur et un mot de passe. Il est impossible de lister dans ce mode d'emploi le nom de l'APN à utiliser avec chaque opérateur, mais si vous faites une recherche sur internet avec le nom de votre opérateur cellulaire et le mot "APN", vous trouverez très facilement le nom de l'APN à utiliser ainsi que le nom d'utilisateur et mot de passe éventuel. Le site apnchanger.org contient ces informations pour les principaux opérateurs de nombreux pays.

Attention ,vous n'avez que trois essais pour configurer le bon code PIN. Passé ces trois essais vous devrez réinitialiser la SIM à l'aide de son code PUK.

Après avoir entré les paramètres de réseau sans fil et éventuellement les avoir testés, vous pouvez cliquer sur le bouton **Ok** pour fermer cette fenêtre de configuration et retourner à la fenêtre de configuration générale.

[2] La fonctionnalité SMS n'est pas encore disponible

Différence entre un réseau GSM et un réseau usuel

Important: Le réseau cellulaire GSM auquel est connecté votre YoctoHub-GSM-3G-NA n'est pas strictement équivalent à une connexion internet usuelle. En effet, l'adresse IP qui attribuée aux modems cellulaires en quasiment toujours une adresse IP *non routée*. Le YoctoHub-GSM-3G-NA voit tout le réseau Internet, mais Internet ne dispose pas d'adresse publique pour le contacter. Ceci signifie que vous ne pourrez pas vous connecter à distance sur votre YoctoHub-GSM-3G-NA depuis n'importe quel ordinateur, juste en tapant son adresse IP dans un navigateur internet.

Une des solution qui permet de palier à ce problème consiste à obtenir de votre opérateur cellulaire une carte SIM permettant une connection directe à travers un réseau privé virtuel.

Utilisation sur un réseau privé virtuel

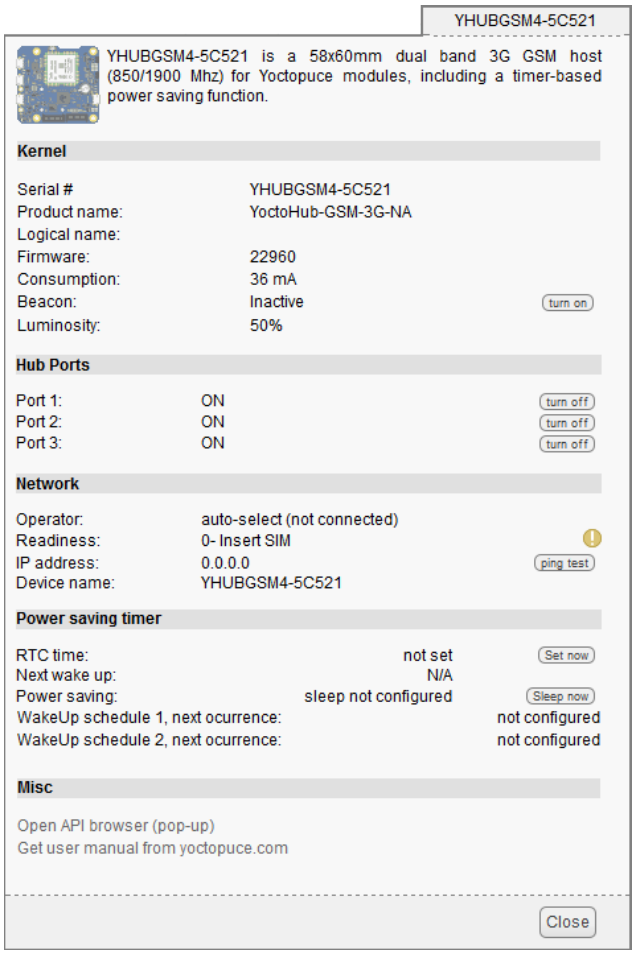
Certains opérateurs cellulaires peuvent fournir sous condition une connection GSM avec un lien internet sur une plage d'adresse privée, qui vous est dédiée. Ce type de connexion, dédié aux services *machine-to-machine*, est généralement réservé aux entreprises. Il permet de se connecter à distance (via un réseau privé virtuel) sur votre YoctoHub-GSM-3G-NA, ce qui n'est autrement pas possible car tout téléphone cellulaire est normalement implicitement isolé derrière un filtre NAT.

Dans le cas ou vous disposez d'une telle connection, vous pouvez configurer quelle adresse IP doit être attribuée au YoctoHub-GSM-3G-NA, et quelle adresse IP est autorisée à le contacter (fonction *firewall*). Pour ce faire, cliquez sur le bouton **edit** en face de la ligne **IP adressing**. Il est indispensable de configurer correctement ces paramètres pour pouvoir contacter votre module par son adresse IP, car sinon le firewall bloquera toute connection entrante.

3.2. Fenêtre d'état du hub

Il est possible de connaitre l'état général du hub c'est à dire don nom logique, l'état du réseau, l'état des ports etc. Pour cela retournez simplement à la liste des modules.

Cliquez sur le numéro de série correspondant à votre YoctoHub-GSM-3G-NA. Cela ouvrira la fenêtre détails de votre module:



Les propriétés du YoctoHub-GSM-3G-NA

Cette fenêtre comporte une section qui relate l'état de la partie réseau du YoctoHub-GSM-3G-NA. Vous y trouverez son adresse MAC, adresse IP courante et nom de réseau. Cette section donne aussi l'état de la connexion réseau. Ces états peuvent être:

- 0- Le module ne trouve pas de réseau GSM (2G). Dans ce cas, vérifiez que:
  - vous avez introduit une carte SIM
  - vous avez configuré le code PIN de la carte SIM dans le YoctoHub-GSM-3G-NA
  - vous n'avez pas désactivé la radio (mode avion)
  - vous avez bien connecté une antenne GSM
  - vous êtes à un endroit où l'on peut recevoir du réseau 2G
- 1- network exists: un réseau cellulaire GSM a été détecté, mais le module n'y est pas encore accepté. Si cet état persiste, vérifiez que votre SIM est valable et que l'opérateur cellulaire choisi correspond.
- 2- network linked: le YoctoHub-GSM-3G-NA a pu se connecter au réseau GSM, mais n'a pas encore établi de connection IP. Si cet état persiste, vérifiez vos réglages APN.
- 3- LAN ready: le réseau local est opérationnel (connection IP avec l'opérateur de téléphonie)
- 4- WWW ready: le module a pu vérifier la connectivité à Internet en se connectant à un serveur de temps (NTP).

Si votre YoctoHub-GSM-3G-NA passe bien à l'état *WWW Ready*, cela signifie que votre connection internet cellulaire fonctionne correctement. Vous pouvez alors passer aux étapes suivantes: connecter des modules Yoctopuce désirés (capteurs et/ou actuateurs), et configurer les interactions avec l'extérieur.

3.3. Configuration automatisée

Il est possible d'industrialiser la configuration réseau du YoctoHub-GSM-3G-NA. Vous trouverez dans les chapitres suivants de cette documentation la description des fonctions de programmation

permettant de relire l'adresse Ethernet d'un module (adresse MAC), et de configurer tous ses paramètres réseau.

Les fonctions de configuration réseau sont aussi accessibles par ligne de commande, en utilisant l'utilitaire `YNetwork` disponible dans la librairie de programmation en ligne de commande[3].

Après avoir effectué un changement de réglage par programmation, prenez garde à appeler la fonction `saveToFlash()` pour vous assurez que les réglages soient sauvés de manière permanente dans la mémoire flash du module.

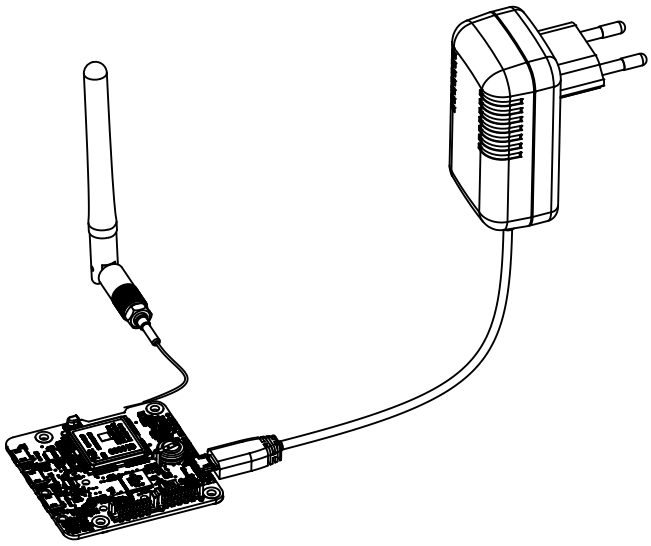
3.4. Connexions

Alimentation

Le YoctoHub-GSM-3G-NA doit être alimenté par la prise USB de contrôle.

USB

Branchez simplement un chargeur USB dans le port *power / control port*, assurez-vous tout de même que le chargeur soit d'une puissance électrique suffisante: le YoctoHub-GSM-3G-NA consomme environ 100mA, auxquels il faudra ajouter la consommation de chaque sous-module. Le YoctoHub-GSM-3G-NA est conçu pour gérer 2A au maximum, c'est pourquoi un chargeur USB capable de délivrer au moins 2A est recommandé. Par ailleurs, vous devrez veiller à ce que la consommation totale de l'ensemble hub + sous-modules ne dépasse pas cette limite.

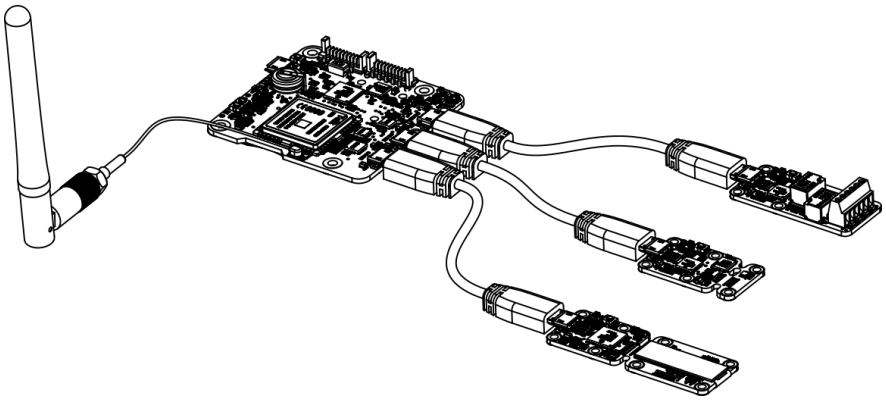


Le YoctoHub-GSM-3G-NA peut être alimenté par un chargeur USB

Sous-modules

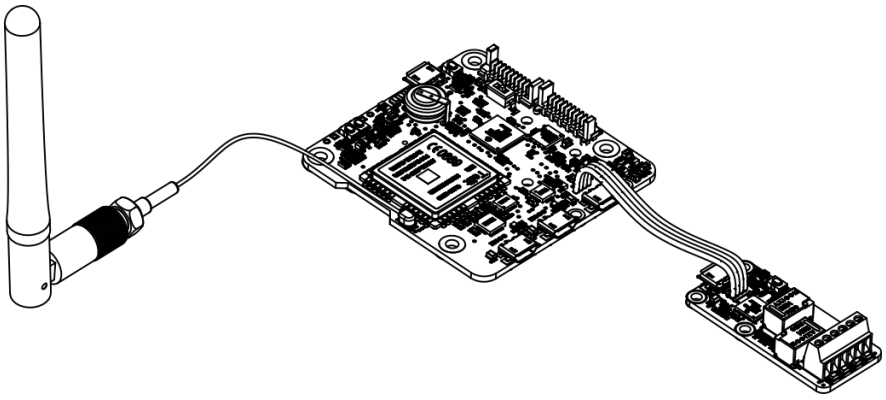
Le YoctoHub-GSM-3G-NA est capable de piloter tous les modules Yoctopuce de la gamme *Yocto*. Ces modules peuvent êtres connectés directement aux ports descendants, ils seront détectés automatiquement. Vous aurez besoin pour cela de câbles USB MicroB-MicroB. Vous pouvez utiliser des câbles OTG ou non, cela n'a pas d'importance.

[3] http://www.yoctopuce.com/FR/libraries.php



Connexion des sous-modules à l'aide de câbles USB

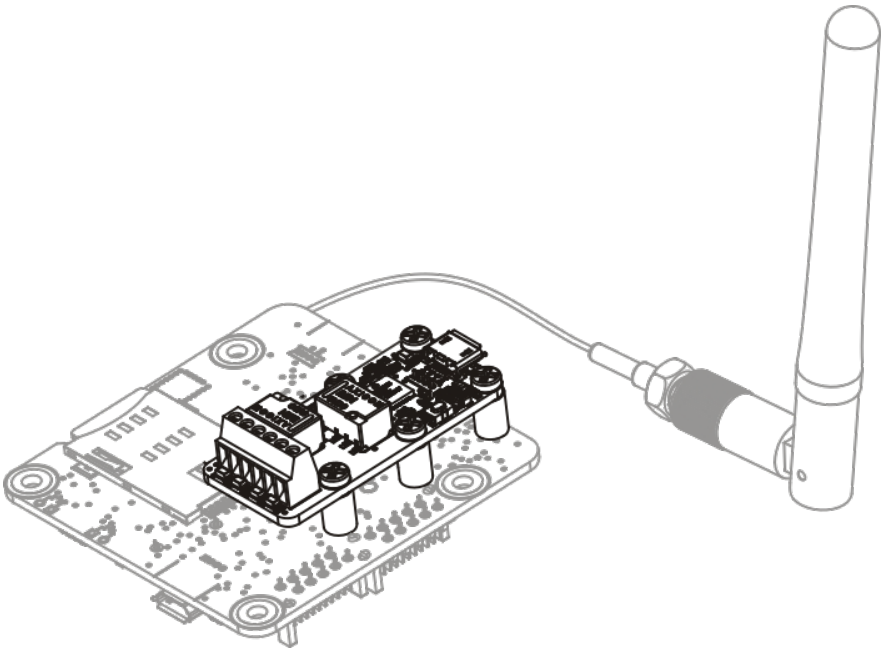
Alternativement, vous pouvez connecter vos modules de manière plus compacte à l'aide de câbles au pas 1.27mm: tous les modules Yoctopuce disposent en effet de contacts à cet effet. Vous pouvez soit souder des connecteurs 1.27mm sur les modules et utiliser des câbles avec connecteurs 1.27mm, soit souder directement du câble plat au pas 1.27mm. Si vous choisissez cette dernière option, il est recommandé d'utiliser du câble plat mono-brin, moins souple que le multi-brin mais beaucoup plus facile à souder. Soyez particulièrement attentif aux polarités: Le YoctoHub-GSM-3G-NA, tout comme l'ensemble de modules de la gamme Yoctopuce, n'est pas protégé contre les inversions de polarité. Une telle inversion a toutes les chances de détruire vos équipements. Assurez-vous que la position du contact carré de part et d'autre du câble correspondent.



Connexion des sous-modules à l'aide de câble nappe

Le YoctoHub-GSM-3G-NA est conçu pour que vous puissiez fixer un module simple largeur directement dessus. Vous aurez besoin de vis, d'entretoises[4] et d'un connecteur au pas 1.27mm[5]. Vous pouvez ainsi transformer un module Yoctopuce USB en en module réseau tout en gardant un format très compact.

[4] http://www.yoctopuce.com/FR/products/accessoires-et-connectique/fix-2-5mm
[5] http://www.yoctopuce.com/FR/products/accessoires-et-connectique/board2board-127



Fixation d'un module directement sur le hub

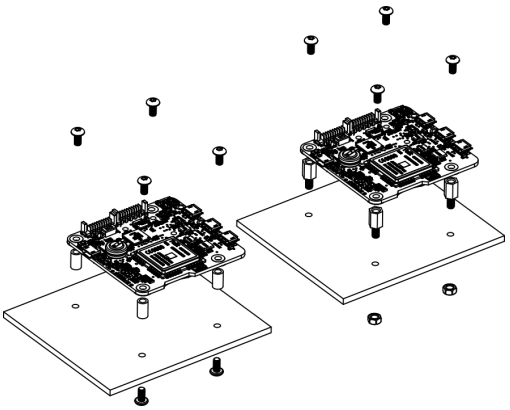
Attention, le YoctoHub-GSM-3G-NA est conçu pour piloter des modules Yoctopuce uniquement. En effet le protocole utilisé entre le YoctoHub-GSM-3G-NA et les sous-modules n'est pas de l'USB mais un protocole propriétaire, beaucoup plus léger. Si d'aventure vous branchez un périphérique autre qu'un module Yoctopuce sur un des ports descendants du YoctoHub-GSM-3G-NA, le port en question sera automatiquement désactivé pour éviter d'endommager le périphérique.

4. Montage

Ce chapitre fournit des explications importantes pour utiliser votre module YoctoHub-GSM-3G-NA en situation réelle. Prenez soin de le lire avant d'aller trop loin dans votre projet si vous voulez éviter les mauvaises surprises.

4.1. Fixation

Pendant la mise au point de votre projet, vous pouvez vous contenter de laisser le hub se promener au bout de son câble. Veillez simplement à ce qu'il ne soit pas en contact avec quoi que soit de conducteur (comme vos outils). Une fois votre projet pratiquement terminé, il faudra penser à faire en sorte que vos modules ne puissent pas se promener à l'intérieur.



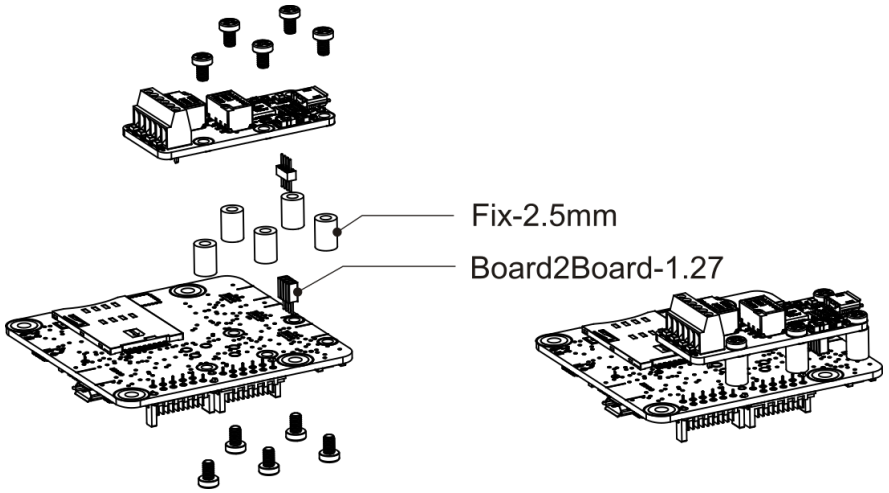
Exemples de montage sur un support.

Le module YoctoHub-GSM-3G-NA dispose de trous de montage 3mm. Vous pouvez utiliser ces trous pour y passer des vis. Le diamètre de la tête de ces vis ne devra pas dépasser 8mm, sous peine d'endommager les circuits du module.

Veillez à que l'électronique module ne soit pas en contact avec le support. La méthode recommandée consiste à utiliser des entretoises. Vous pouvez monter le module dans le sens qui vous convient: mais vous devez conscient du fait que les composants électroniques du YoctoHub-GSM-3G-NA, la partie réseau en particulier, dégagent de la chaleur. Vous devrez donc faire en sorte que la chaleur ne puisse pas s'accumuler.

4.2. Fixation d'un sous-module

Le YoctoHub-GSM-3G-NA est conçu pour que vous puissiez visser un module simple largeur directement dessus. Par simple largeur, on entend les modules de 20 mm de large. Tous les modules simple largeur ont leurs 5 trous de fixation et le connecteur USB au même endroit. Le sous-module peut être fixé à l'aide de vis et d'entretoises. Il y a derrière les connecteurs USB du YoctoHub-GSM-3G-NA et du sous-module un ensemble de 4 contacts qui permettent d'effectuer la connexion électrique entre le hub et le sous-module. Si vous ne vous sentez pas suffisamment à l'aise avec un fer à souder, vous pouvez aussi aussi utiliser un simple câble USB MicroB-MicroB, OTG ou non.



Fixation d'un module directement sur le hub

Prenez garde à bien monter le module sur la face prévue, comme illustré ci-dessus. Les 5 trous du module doivent correspondre aux 5 trous du YoctoHub-GSM-3G-NA, et le contact carré sur sur le module doit être connecté au contact carré sur le port descendant du YoctoHub-GSM-3G-NA. Si vous montez un module sur l'autre face ou d'une autre manière, la polarité du connecteur sera inversée et vous risquez fort d'endommager définitivement votre matériel.

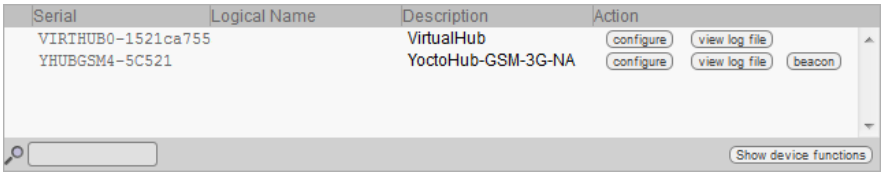
Tous les accessoires nécessaires à la fixation d'un module sur votre YoctoHub-GSM-3G-NA sont relativement courants. Vous pourrez les trouver sur le site de Yoctopuce tout comme sur la plupart des sites vendant du matériel électronique. Attention cependant, la tête des vis servant à fixer le sous-module devra avoir un diamètre maximum de 4.5 millimètres, sous peine d'endommager les composants électroniques.

5. Interactions avec l'extérieur

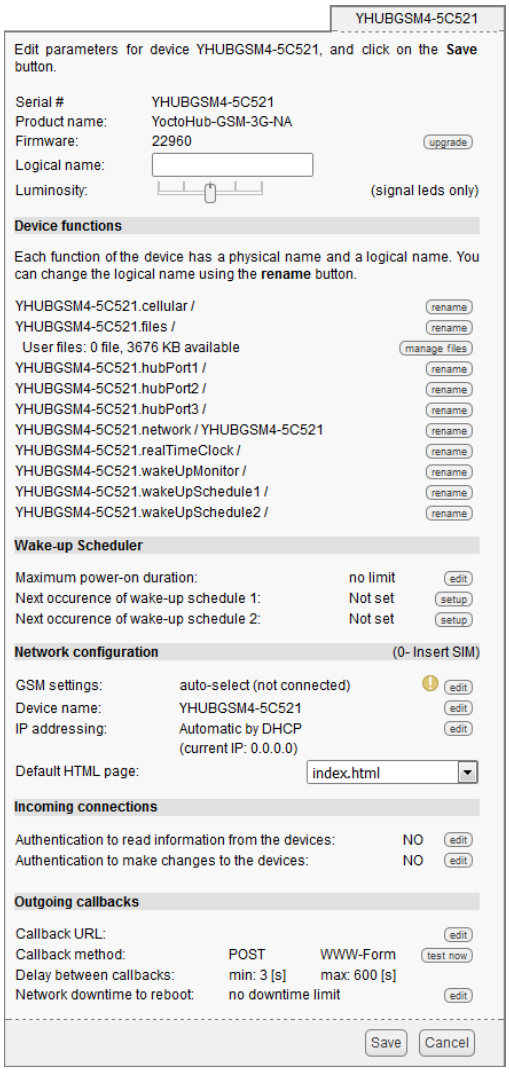
Le YoctoHub-GSM-3G-NA est capable de poster sur le site web de votre choix l'état des modules qu'il voit. Les valeurs sont postées à intervalles réguliers et à chaque fois qu'une valeur change de manière significative. Cette fonctionnalité, appelée *Callback HTTP*, vous permettra d'interfacer vos modules Yoctopuce avec divers services web.

5.1. Configuration

Pour utiliser cette fonctionnalité, cliquez simplement sur le bouton **Configure** de la ligne correspondant au YoctoHub-GSM-3G-NA dans l'interface, puis cliquez sur le bouton **edit** de la section **Outgoing callback**.



Cliquez sur le bouton "Configure" de la première ligne



Puis éditez la section Outgoing callbacks.

La fenêtre de configuration des callbacks HTTP apparaît. Cette fenêtre va vous permettre de définir comment votre YoctoHub-GSM-3G-NA va interagir avec un serveur Web externe. Vous avez plusieurs type d'interactions a votre disposition. Pour chaque type, un wizard vous guidera pour fournir les paramètres adéquats.

5.2. Emoncms

Emoncms.org est un service de cloud open-source gratuit où vous pouvez vous inscrire pour poster les données de vos capteurs, et qui vous permettra de les visualiser en temps réel par Internet et de tracer des graphiques d'historique, et ce sans écrire la moindre ligne de code. Il suffit d'indiquer dans le fenêtre de configuration votre clé d'API fournie par Emoncms, ainsi que le numéro de noeud (arbitraire) que vous désirez attribuer à votre YoctoHub-GSM-3G-NA.

Il est aussi possible d'installer Emoncms sur votre propre serveur, afin de garder le contrôle sur vos données. Vous trouverez des explications plus détaillées sur le blog de Yoctopuce[1].

Yoctopuce n'est en aucune manière affilié à Emoncms.org.

[1] http://www.yoctopuce.com/FR/article/utiliser-emoncms-sur-un-serveur-prive

5.3. Valarm.net

Valarm est un service de cloud professionnel où vous pouvez vous inscrire pour poster les données de vos capteurs, et qui offre des fonctions avancées notemment pour la configuration à distance des modules Yoctopuce et la géolocalisation des mesures.

Valarm est revendeur des produits Yoctopuce, mais Yoctopuce n'est pas autrement affilié à Valarm.

5.4. Xively (anciennement Cosm)

Xively est un service de cloud payant où vous pourriez poster les données de vos capteurs. Notez que depuis la fin 2015, Xively concentre son activité sur les grosses entreprises et les clients OEM. De ce fait, le service Xively n'est plus forcément disponible pour n'importe qui. Pour plus de détails, consultez xively.com.

Yoctopuce n'est en aucune manière affilié à Xively.

5.5. InfluxDB

InfluxDB est une base de données dédiée spécifiquement à stocker des séries temporelles de mesures et d'événements. Elle est très efficace pour retrouver des séries de mesures pour une plage de temps donnée, y compris en les rassemblant pour en faire un résumé. Vous pouvez facilement l'installer sur votre propre ordinateur pour enregistrer vos données, et dessiner des graphiques. Vous trouverez un guide pas-à-pas pour configurer InfluxDB et Grafana avec les modules Yoctopuce sur le blog de Yoctopuce. [2].

Yoctopuce n'est en aucune manière affilié à InfluxData ni à Grafana.

5.6. PRTG

PRTG est une solution commerciale développée par PAESSLER, destinée à la supervision des systèmes et des applications. Vous pouvez facilement l'installer sur Windows pour enregistrer les mesures et obtenir des graphiques de vos capteurs. Pour plus de détails, voir fr.paessler.com/prtg. Vous trouverez un guide pas-à-pas pour configurer PRTG avec les modules Yoctopuce sur le blog de Yoctopuce [3].

Yoctopuce n'est en aucune manière affilié à PAESSLER.

5.7. MQTT

MQTT est un protocole de l'Internet des Objets permettant à des capteurs de publier en temps réel des valeurs vers un serveur central, appelé broker MQTT. Pour plus de détails, voir mqtt.org. Vous trouverez plusieurs articles sur le blog de Yoctopuce décrivant l'utilisation de MQTT.

5.8. Yocto-API callback

Avec certains langages de programmation, l'API Yoctopuce est capable de fonctionner en mode *callback HTTP*. Dans ce mode un script sur le serveur web peut prendre le contrôle de vos modules à travers un filtre NAT sans que vous ayez à ouvrir un port. Typiquement cela permet de contrôler depuis un site Web public des modules Yoctopuce installés derrière un router DSL privé. Le YoctoHub-GSM-3G-NA sert alors de passerelle. Vous avez simplement à définir l'URL du script de contrôle sur le serveur HTTP et éventuellement les crédentiels nécessaires pour y accéder. Dans le script serveur, vous devez alors initialiser la librairie Yoctopuce avec l'appel suivant:

[2] http://www.yoctopuce.com/FR/article/utiliser-les-capteurs-yoctopuce-avec-influxdb-et-grafana
[3] http://www.yoctopuce.com/FR/article/nouveaute-le-support-prtg-dans-les-yoctohub

```
RegisterHub("http://callback");
```

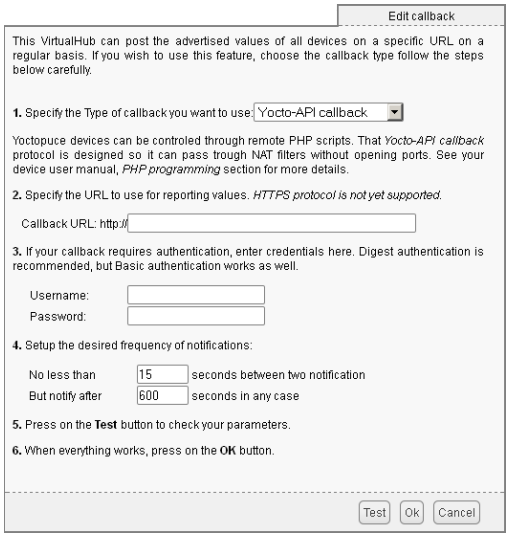
Il existe deux possibilités pour utiliser l'API Yoctopuce enmode callback. La première, disponible en PHP, Java et Node.JS est basée sur des connections en pur HTTP. Le YoctoHub-GSM-3G-NA poste l'intégralité de son état au serveur, et reçoit en retour des commandes du script sur le serveur. Ce fonctionnement implique quelques limitations: les intéractions complexes, comme la récupération de données depuis l'enregistreur de données sur les capteurs, ne sont pas possible.

Le deuxième mode fonctionnement de l'API par callback est l'utilisation des callbacks WebSocket. Ce mode est disponible uniquement avec les serveurs Java et Node.JS. Les WebSockets sont une extension standard de HTTP, qui permet l'établissement d'une liaison bidirectionnelle complète sur une liaison HTTP. Lorsqu'un script serveur est connecté à un YoctoHub-GSM-3G-NA au travers d'un callback WebSocket, l'API Yoctopuce peut être utilisée sans la moindre limitation.

L'application web **GatewayHub**, disponible sur le site de Yoctopuce, utilise cette technologie de callback Websocket pour permettre un accès à distance au YoctoHub-GSM-3G-NA, même à travers un filtre NAT ou un pare-feu. Pour plus d'informations, référez-vous au blog de Yoctopuce [4].

5.9. User defined callback

Les "*User defined callback*" vous permettent de personnaliser la manière dont votre YoctoHub-GSM-3G-NA va interagir avec un site Web externe. Vous avez besoin de définir l'URL du serveur Web sur lequel le YoctoHub-GSM-3G-NA va poster l'état de ses devices. Notez que seul le protocole HTTP est supporté (pas de HTTPS).



La fenêtre de configurations des callbacks

Si vous désirez protéger votre script de callback, vous pouvez configurer un contrôle d'accès HTTP standard sur le serveur Web. Le YoctoHub-GSM-3G-NA sait comment gérer les méthodes standard d'identification de HTTP: indiquez simplement le nom d'utilisateur et le mot de passe nécessaires pour accéder à la page. Il est possible d'utiliser la méthode "Basic" aussi bien que la méthode "Digest", mais il est recommandé d'utiliser la méthode "Digest", car elle est basée sur un protocole de question-réponse qui évite la transmission du mot de passe sur le réseau et évite aussi les copies d'autorisation.

Le YoctoHub-GSM-3G-NA poste avec la méthode POST les valeurs notifiées[5] des modules à intervalle régulier, et à chaque fois qu'une de ces valeurs change de manière significative. Vous pouvez changer les délais d'attente entre les posts.

[4] http://www.yoctopuce.com/FR/article/une-passerelle-pour-acceder-aux-yoctohubs-a-distance
[5] Les valeurs notifiées sont celles que vous voyez quand vous cliquez sur *show functions* dans l'interface principale du YoctoHub-GSM-3G-NA .

Tests

Afin de vous permettre de déboguer le processus, le YoctoHub-GSM-3G-NA vous permet de visualiser la réponse au callback envoyé par le serveur Web. Cliquez simplement sur le bouton **test** une fois que vous avez renseigné tous les champs. Si le résultat vous paraît satisfaisant, fermez la fenêtre de debug, et cliquez sur **Ok**.

Formats

Les valeurs sont postées sous une des formes suivantes:

1. Si un nom logique a été défini pour une fonction:

```
NOM_LOGIQUE_DE_LA_FONCTION = VALEUR
```

2. Si un nom logique a été défini pour le module, mais pas pour la fonction:

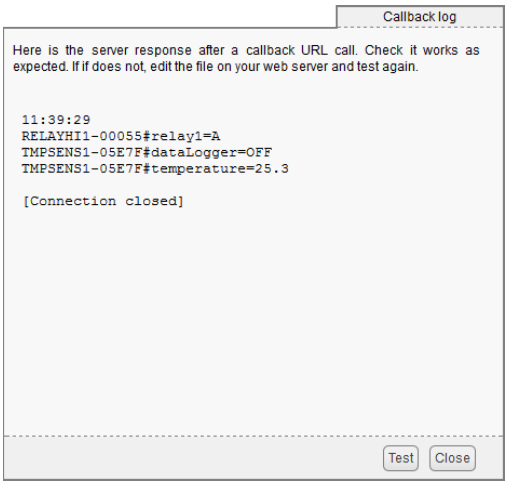
```
NOM_DU_MODULE#NOM_HARDWARE = VALUE
```

3. Si aucun nom logique n'a été attribué:

```
NUMERO_DE_SERIE#NOM_HARDWARE = VALEUR
```

Voici un script PHP qui vous permettra de visualiser le contenu des données postées par le callback, suivi du résultat dans la fenêtre de debug.

```
<?php
  Print(Date('H:i:s')."\r\n");
  foreach ($_POST as $key=>$value) {
      Print("$key=$value\r\n");
  }
?>
```



Le résultat du test de callback avec un Yocto-PowerRelay et un Yocto-Temperature.

6. Programmation

6.1. Accès aux modules connectés

Le YoctoHub-GSM-3G-NA se comporte exactement comme un ordinateur faisant tourner un *VirtualHub*. La seule différence entre un programme utilisant l'API Yoctopuce utilisant des modules en USB natif et ce même programme utilisant des modules Yoctopuce connecté à un YoctoHub-GSM-3G-NA se situe au niveau de l'appel a *registerHub*. Pour utiliser des modules USB connectés en natif, le paramètre de *RegisterHub* est *usb*, pour utiliser des modules connectés à un YoctoHub-GSM-3G-NA, il suffit de remplacer ce paramètre par l'adresse IP du YoctoHub-GSM-3G-NA. Par en exemple, en Delphi:

```
YRegisterHub("usb",errmsg);
```

devient

```
YRegisterHub("192.168.0.10",errmsg); // l'adresse IP du hub est 192.168.0.10
```

6.2. Contrôle du YoctoHub-GSM-3G-NA

Du point de vue API de programmation, le YoctoHub-GSM-3G-NA est un module comme les autres. Il est parfaitement contrôlable depuis l'API Yoctopuce. Pour ce faire, vous aurez besoin des classes suivantes.

Module

Cette classe, commune à tous les modules Yoctopuce permet de contrôler le module en temps que tel. Elle vous permettra de contrôler la Yocto-Led, de connaître la consommation sur USB du YoctoHub-GSM-3G-NA, etc.

Cellular

Cette classe permet de contrôler la configuration du réseau cellulaire du YoctoHub-GSM-3G-NA, en particulier le nom de l'opérateur cellulaire, le code PIN pour utiliser la carte SIM et les paramètres de l'APN.

Network

Cette classe permet de contrôler la partie réseau du YoctoHub-GSM-3G-NA, vous pourrez contrôler l'état du link, lire l'adresse MAC, changer l'adresse IP du YoctoHub-GSM-3G-NA, connaître la consommation sur PoE, etc.

HubPort

Cette classe permet de contrôler la partie hub. Vous pourrez activer ou désactiver les ports du YoctoHub-GSM-3G-NA, vous pourrez aussi savoir quel module est connecté à quel port.

Files

Cette classe permet d'accéder aux fichiers stockées dans la mémoire flash du YoctoHub-GSM-3G-NA. Le YoctoHub-GSM-3G-NA dispose en effet d'un petit système de fichiers qui vous permet de stocker par exemple une Web App contrôlant les modules connectés au YoctoHub-GSM-3G-NA.

WakeMonitor

Cette classe permet de contrôler la mise en sommeil du YoctoHub-GSM-3G-NA.

WakeSchedule

Cette classe permet d'agender un ou plusieurs réveils du YoctoHub-GSM-3G-NA.

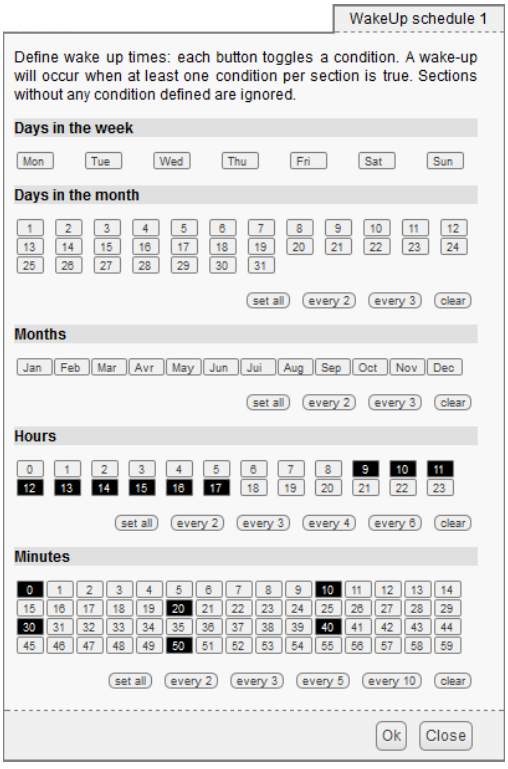
Vous trouverez quelques exemples de contrôle du YoctoHub-GSM-3G-NA par programmation dans les librairies Yoctopuce, disponibles gratuitement sur le site de Yoctopuce.

7. Mise en sommeil

Le YoctoHub-GSM-3G-NA dispose d'une horloge en temp réel (RTC) alimentée par un super condensateur, qui se recharge automatiquement lorsque le module est sous tension mais permet de maintenir l'heure sans aucune alimentation pendant plusieurs jours. Ce RTC est utilisé pour piloter un système de mise en sommeil afin d'économiser l'énergie. Le système de mise en sommeil peut être configuré manuellement à l'aide d'une interface, ou piloté par logiciel.

7.1. Configuration manuelle du système de réveil

Les conditions de réveil peuvent être configurées manuellement en vous connectant sur l'interface du YoctoHub-GSM-3G-NA. Dans la section **Wake-up scheduler** de la fenêtre de configuration générale, cliquez sur le bouton setup correspondant à l'un des "wakeup-schedule". Une fenêtre qui permet d'agender des réveils plus ou moins réguliers s'ouvre. Il suffit de sélectionner les cases correspondant aux occurrences désirées. Les sections laissées vides seront ignorée.



Fenêtre de configuration des réveils, ici toutes les 10 minutes entre 9h et 17h

De même, vous pouvez configurer directement sur l'interface du YoctoHub-GSM-3G-NA le temps d'éveil maximal désiré, après lequel le module retournera automatiquement en sommeil profond. Si vous utilisez votre YoctoHub-GSM-3G-NA sur batteries, ceci vous assurera de ne pas vider les batteries même si aucun ordre de mise en sommeil explicite n'est reçu.

7.2. Paramétrage du système de réveil par logiciel

Au niveau de l'interface de programmation, le système de réveil est implémenté a l'aide de deux types de fonction : La fonction *wakeUpMonitor* et la fonction *WakeUpSchedule*.

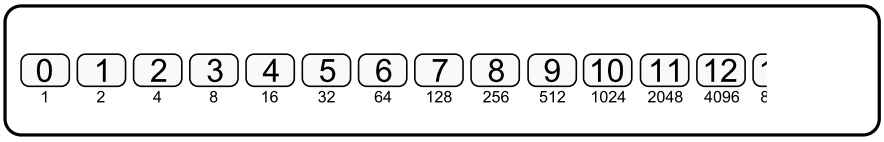
wakeUpMonitor

La fonction *wakeUpMonitor* gère l'éveil et la mise en sommeil proprement dits. Elle met à disposition toutes les fonctionnalités de contrôle immédiate: éveil immédiat, mise en sommeil immédiate, calcul de la date du prochain réveil etc... Le fonction *wakeUpMonitor* permet aussi de définir le temps maximum pendant lequel le YoctoHub-GSM-3G-NA peut rester éveil avant de de se mettre automatiquement en sommeil.

wakeUpSchedule

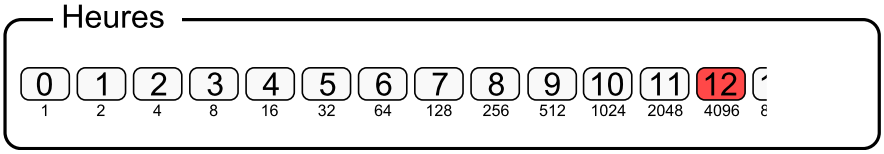
La fonction *wakeUpSchedule* permet de programmer une condition de réveil. Elle dispose de cinq variables qui permet de définir des correspondance sur les minutes, heure, jour de la semaine, jour dans le mois, et mois. Ces variables sont des variables entières dont chaque bit défini une correspondance. Schématiquement, chaque ensemble de minutes, jours, heures est représenté sous la forme d'un ensemble de case avec chacune un coefficient qui est une puissante de deux, exactement comme dans l'interface correspondante du YoctoHub-GSM-3G-NA.

Par exemple le bit 0 des heures correspond à l'heure zéro, le bit 1 correspond à l'heure une, le bit 2 correspond à l'heure 2 etc.



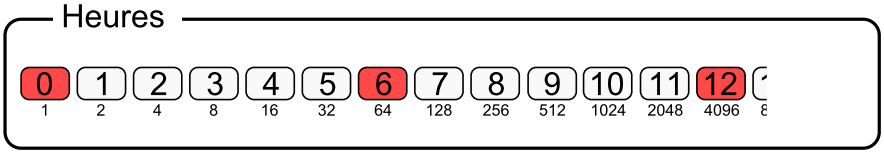
Chaque case se voit affecter une puissante de deux

Ainsi pour programmer le YoctoHub-GSM-3G-NA pour qu'il se réveille tout les jours a midi, il mettre le bit 12 à 1, ce qui correspond à la valeur 2^12 = 4096.



Exemple de réveil à 12 H

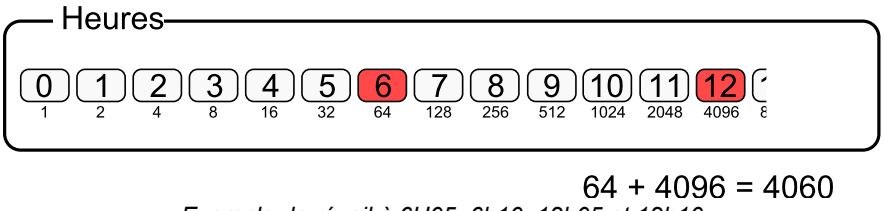
Pour que le module se reveille à 0 heure, 6 heures et 12 heures, il faut mettre les bit 0,6,et 12 à un, ce qui correspondant à la valeur 2^0+ 2^6 + 2^12 = 1 + 64 + 4096 = 4161



1 + 64 + 4096 = 4151

Exemple de réveil à 0, 6 et 12 H

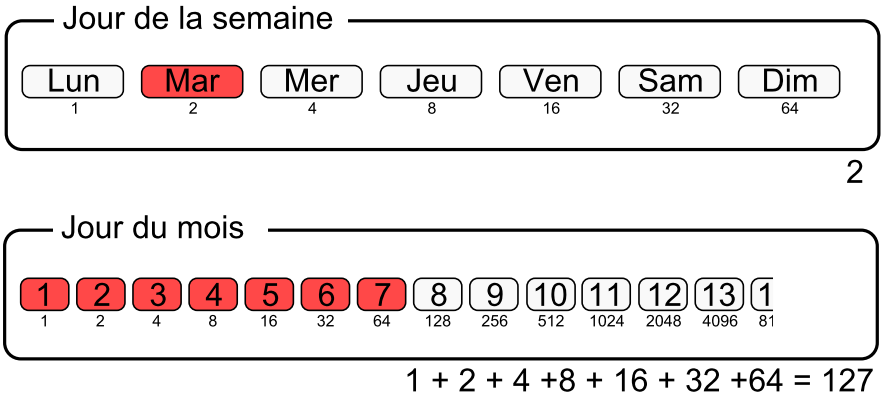
Les variables peuvent être combinées, pour qu'un réveil ait lieu tous les jours à 6H05, 6h10, 12h05 et 12h10 il suffit de mettre les heures à 2^6 + 2^12 = 4060 et les minutes à 2^5 + 2^10= 1056. Les variables laissée à zéro sont ignorées.



Exemple de réveil à 6H05, 6h10, 12h05 et 12h10

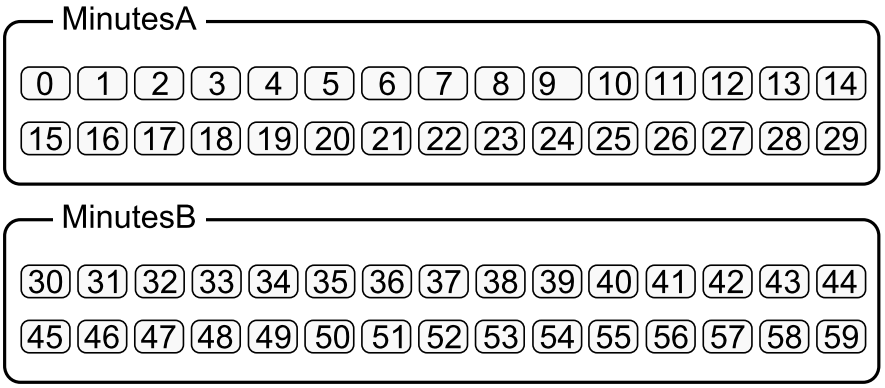
Notez que si vous désirez programmer un réveil à 6H05 et 12h10, mais pas 6h10 et 12h10, vous aurez besoin d'utiliser deux fonctions *wakeUpSchedule* différentes.

Ce paradigme permet de programmer des réveils assez complexe. Ainsi pour programmer un réveil tous les premiers mardis du mois, faut mettre à un le deuxième bit des jours de la semaine et les sept premiers bit des jours du mois.



Exemple de réveil tous les premiers mardi du mois

Certains languages de programmation, dont Javascript et Python, ne supportent pas les entiers 64 bits, ce qui pose un problème pour encoder les minutes. C'est pourquoi les minutes sont à la fois accessibles via un entier 64 bits *minutes* et deux entiers 32 bits, *minutesA* et *minutesB*, qui eux sont disponibles dans tous les languages actuels.



Les minutes sont aussi disponibles sous forme de deux entiers 32 bits.

La fonction *wakeUpSchedule* dispose d'une variable supplémentaire qui permet de définir le temps, en secondes, durant lequel le module restera éveillé après un réveil. Si cette variable est mise à zéro, le module restera éveillé.

Le YoctoHub-GSM-3G-NA dispose de deux fonctions *wakeUpSchedule* ce qui permet de programmer jusqu'a deux types de réveils indépendants.

8. Référence de l'API de haut niveau

Ce chapitre résume les fonctions de l'API de haut niveau pour commander votre YoctoHub-GSM-3G-NA. La syntaxe et les types précis peuvent varier d'un langage à l'autre mais, sauf avis contraire toutes sont disponibles dans chaque language. Pour une information plus précise sur les types des arguments et des valeurs de retour dans un langage donné, veuillez vous référer au fichier de définition pour ce langage (`yocto_api.*` ainsi que les autres fichiers `yocto_*` définissant les interfaces des fonctions).

Dans les langages qui supportent les exceptions, toutes ces fonctions vont par défaut générer des exceptions en cas d'erreur plutôt que de retourner la valeur d'erreur documentée pour chaque fonction, afin de faciliter le déboguage. Il est toutefois possible de désactiver l'utilisation d'exceptions à l'aide de la fonction `yDisableExceptions()`, si l'on préfère travailler avec des valeurs de retour d'erreur.

Ce chapitre ne reprend pas en détail les concepts de programmation des modules Yoctopuce. Vous trouverez des explications plus détaillées dans la documentation des modules que vous souhaitez raccorder à votre YoctoHub-GSM-3G-NA.

8.1. La classe YHubPort

Interface pour intéragir avec les ports de YoctoHub, disponibles par exemple dans le YoctoHub-Ethernet, le YoctoHub-GSM-3G-EU, le YoctoHub-Shield et le YoctoHub-Wireless-g

La classe `YHubPort` permet de contrôler l'alimentation des ports descendants d'un YoctoHub. Il permet de détecter si un module y est raccordé et lequel. Un YHubPort reçoit toujours automatiquement comme nom logique le numéro de série unique du module Yoctopuce qui y est connecté.

Pour utiliser les fonctions décrites ici, vous devez inclure:

es	in HTML: <script src="../../lib/yocto_hubport.js"></script> in node.js: require('yoctolib-es2017/yocto_hubport.js');
js	<script type='text/javascript' src='yocto_hubport.js'></script>
cpp	#include "yocto_hubport.h"
m	#import "yocto_hubport.h"
pas	uses yocto_hubport;
vb	yocto_hubport.vb
cs	yocto_hubport.cs
dnp	import YoctoProxyAPI.YHubPortProxy
java	import com.yoctopuce.YoctoAPI.YHubPort;
uwp	import com.yoctopuce.YoctoAPI.YHubPort;
py	from yocto_hubport import *
php	require_once('yocto_hubport.php');
vi	YHubPort.vi

Fonction globales

YHubPort.FindHubPort(**func**)

Permet de retrouver un port de YoctoHub d'après un identifiant donné.

YHubPort.FindHubPortInContext(**yctx**, **func**)

Permet de retrouver un port de YoctoHub d'après un identifiant donné dans un Context YAPI.

YHubPort.FirstHubPort()

Commence l'énumération des ports de YoctoHub accessibles par la librairie.

YHubPort.FirstHubPortInContext(**yctx**)

Commence l'énumération des ports de YoctoHub accessibles par la librairie.

YHubPort.GetSimilarFunctions()

Enumère toutes les fonctions de type HubPort disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).

Propriétés des objets `YHubPortProxy`

hubport→AdvertisedValue *[lecture seule]*

Courte chaîne de caractères représentant l'état courant de la fonction.

hubport→Enabled *[modifiable]*

Vrai si le port du YoctoHub est alimenté, faux sinon.

hubport→FriendlyName *[lecture seule]*

Identifiant global de la fonction au format `NOM_MODULE.NOM_FONCTION`.

hubport→FunctionId *[lecture seule]*

Identifiant matériel du port de YoctoHub, sans référence au module.

hubport→HardwareId *[lecture seule]*

Identifiant matériel unique de la fonction au format SERIAL.FUNCTIONID.

hubport→IsOnline *[lecture seule]*

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

hubport→LogicalName *[modifiable]*

Nom logique de la fonction.

hubport→PortState *[lecture seule]*

état actuel du port de YoctoHub.

hubport→SerialNumber *[lecture seule]*

Numéro de série du module, préprogrammé en usine.

Méthodes des objets YHubPort

hubport→clearCache()

Invalide le cache.

hubport→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du port de YoctoHub au format TYPE(NAME)=SERIAL.FUNCTIONID.

hubport→get_advertisedValue()

Retourne la valeur courante du port de YoctoHub (pas plus de 6 caractères).

hubport→get_baudRate()

Retourne la vitesse de transfert utilisée par le port de YoctoHub, en kbps.

hubport→get_enabled()

Retourne vrai si le port du YoctoHub est alimenté, faux sinon.

hubport→get_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du port de YoctoHub.

hubport→get_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du port de YoctoHub.

hubport→get_friendlyName()

Retourne un identifiant global du port de YoctoHub au format NOM_MODULE.NOM_FONCTION.

hubport→get_functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

hubport→get_functionId()

Retourne l'identifiant matériel du port de YoctoHub, sans référence au module.

hubport→get_hardwareId()

Retourne l'identifiant matériel unique du port de YoctoHub au format SERIAL.FUNCTIONID.

hubport→get_logicalName()

Retourne le nom logique du port de YoctoHub.

hubport→get_module()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

hubport→get_module_async(**callback**, **context**)

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

hubport→get_portState()

Retourne l'état actuel du port de YoctoHub.

hubport→get_serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

hubport→get_userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

hubport→isOnline()

Vérifie si le module hébergeant le port de YoctoHub est joignable, sans déclencher d'erreur.

hubport→isOnline_async(**callback**, **context**)

Vérifie si le module hébergeant le port de YoctoHub est joignable, sans déclencher d'erreur.

hubport→isReadOnly()

Test si la fonction est en lecture seule.

hubport→load(**msValidity**)

Met en cache les valeurs courantes du port de YoctoHub, avec une durée de validité spécifiée.

hubport→loadAttribute(**attrName**)

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

hubport→load_async(**msValidity**, **callback**, **context**)

Met en cache les valeurs courantes du port de YoctoHub, avec une durée de validité spécifiée.

hubport→muteValueCallbacks()

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

hubport→nextHubPort()

Continue l'énumération des ports de YoctoHub commencée à l'aide de `yFirstHubPort()` Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les ports de YoctoHub sont retournés.

hubport→registerValueCallback(**callback**)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

hubport→set_enabled(**newval**)

Modifie le mode d'activation du port du YoctoHub.

hubport→set_logicalName(**newval**)

Modifie le nom logique du port de YoctoHub.

hubport→set_userData(**data**)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

hubport→unmuteValueCallbacks()

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

hubport→wait_async(**callback**, **context**)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

YHubPort.FindHubPort() YHubPort
YHubPort.FindHubPort()

Permet de retrouver un port de YoctoHub d'après un identifiant donné.

```
js      function yFindHubPort( func)
cpp     YHubPort* yFindHubPort( string func)
m       +(YHubPort*) FindHubPort : (NSString*) func
pas     TYHubPort yFindHubPort( func: string): TYHubPort
vb      function yFindHubPort( ByVal func As String) As YHubPort
cs      static YHubPort FindHubPort( string func)
dnp     static YHubPortProxy FindHubPort( string func)
java    static YHubPort FindHubPort( String func)
uwp     static YHubPort FindHubPort( string func)
py      FindHubPort( func)
php     function yFindHubPort( $func)
es      static FindHubPort( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le port de YoctoHub soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YHubPort.isOnline()` pour tester si le port de YoctoHub est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Si un appel à la méthode is_online() de cet objet renvoie FAUX alors que vous êtes sûr que le module correspondant est bien branché, vérifiez que vous n'avez pas oublié d'appeler registerHub() à l'initialisation de de l'application.

Paramètres :

func une chaîne de caractères qui référence le port de YoctoHub sans ambiguïté, par exemple `YHUBETH1.hubPort1`.

Retourne :

un objet de classe `YHubPort` qui permet ensuite de contrôler le port de YoctoHub.

YHubPort.FindHubPortInContext() YHubPort
YHubPort.FindHubPortInContext()

Permet de retrouver un port de YoctoHub d'après un identifiant donné dans un Context YAPI.

java static YHubPort **FindHubPortInContext**(YAPIContext **yctx**, String **func**)
uwp static YHubPort **FindHubPortInContext**(YAPIContext **yctx**, string **func**)
es static **FindHubPortInContext**(**yctx**, **func**)

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le port de YoctoHub soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YHubPort.isOnline()` pour tester si le port de YoctoHub est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Paramètres :

yctx un contexte YAPI

func une chaîne de caractères qui référence le port de YoctoHub sans ambiguïté, par exemple `YHUBETH1.hubPort1`.

Retourne :

un objet de classe `YHubPort` qui permet ensuite de contrôler le port de YoctoHub.

YHubPort.FirstHubPort() YHubPort
YHubPort.FirstHubPort()

Commence l'énumération des ports de YoctoHub accessibles par la librairie.

js	function **yFirstHubPort**( )
cpp	YHubPort * **yFirstHubPort**( )
m	+(YHubPort*) **FirstHubPort**
pas	TYHubPort **yFirstHubPort**( ): TYHubPort
vb	function **yFirstHubPort**( ) As YHubPort
cs	static YHubPort **FirstHubPort**( )
java	static YHubPort **FirstHubPort**( )
uwp	static YHubPort **FirstHubPort**( )
py	**FirstHubPort**( )
php	function **yFirstHubPort**( )
es	static **FirstHubPort**( )

Utiliser la fonction `YHubPort.nextHubPort()` pour itérer sur les autres ports de YoctoHub.

Retourne :

un pointeur sur un objet `YHubPort`, correspondant au premier port de YoctoHub accessible en ligne, ou `null` si il n'y a pas de ports de YoctoHub disponibles.

YHubPort.FirstHubPortInContext() YHubPort
YHubPort.FirstHubPortInContext()

Commence l'énumération des ports de YoctoHub accessibles par la librairie.

java static YHubPort **FirstHubPortInContext**(YAPIContext **yctx**)
uwp static YHubPort **FirstHubPortInContext**(YAPIContext **yctx**)
es static **FirstHubPortInContext**(**yctx**)

Utiliser la fonction `YHubPort.nextHubPort()` pour itérer sur les autres ports de YoctoHub.

Paramètres :

yctx un contexte YAPI.

Retourne :

un pointeur sur un objet `YHubPort`, correspondant au premier port de YoctoHub accessible en ligne, ou `null` si il n'y a pas de ports de YoctoHub disponibles.

YHubPort.GetSimilarFunctions() YHubPort
YHubPort.GetSimilarFunctions()

Enumère toutes les fonctions de type HubPort disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).

dnp static new string[] **GetSimilarFunctions**( )

Chaque chaîne retournée peut être passée en argument à la méthode `YHubPort.FindHubPort` pour obtenir une objet permettant d'intéragir avec le module correspondant.

Retourne :

un tableau de chaînes de caractères, contenant les identifiants matériels de chaque fonction disponible trouvée.

hubport→AdvertisedValue — YHubPort

Courte chaîne de caractères représentant l'état courant de la fonction.

dnp string **AdvertisedValue**

hubport→Enabled — YHubPort

Vrai si le port du YoctoHub est alimenté, faux sinon.

```
dnp  int Enabled
```

Valeurs possibles:

```
Y_ENABLED_INVALID = 0
Y_ENABLED_FALSE   = 1
Y_ENABLED_TRUE    = 2
```

Modifiable. Modifie le mode d'activation du port du YoctoHub. Si le port est actif, il sera alimenté. Sinon, l'alimentation du module est coupée.

hubport→FriendlyName — YHubPort

Identifiant global de la fonction au format NOM_MODULE.NOM_FONCTION.

dnp string **FriendlyName**

Le chaîne retournée utilise soit les noms logiques du module et de la fonction si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel de la fonction (par exemple: MyCustomName.relay1)

hubport→FunctionId — YHubPort

Identifiant matériel du port de YoctoHub, sans référence au module.

dnp string **FunctionId**

Par example `relay1`.

hubport→HardwareId — YHubPort

Identifiant matériel unique de la fonction au format SERIAL.FUNCTIONID.

dnp string **HardwareId**

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de la fonction (par example RELAYLO1-123456.relay1).

hubport→IsOnline YHubPort

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

dnp bool **IsOnline**

Si les valeurs des attributs en cache de la fonction sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

hubport→LogicalName — YHubPort

Nom logique de la fonction.

dnp string **LogicalName**

Modifiable. Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

hubport→PortState — YHubPort

état actuel du port de YoctoHub.

dnp int **PortState**

Valeurs possibles:

Y_PORTSTATE_INVALID	= 0
Y_PORTSTATE_OFF	= 1
Y_PORTSTATE_OVRLD	= 2
Y_PORTSTATE_ON	= 3
Y_PORTSTATE_RUN	= 4
Y_PORTSTATE_PROG	= 5

hubport→SerialNumber YHubPort

Numéro de série du module, préprogrammé en usine.

dnp string **SerialNumber**

hubport→clearCache() YHubPort

Invalide le cache.

js	function **clearCache**( )
cpp	void **clearCache**( )
m	-(void) **clearCache**
pas	**clearCache**( )
vb	procedure **clearCache**( )
cs	void **clearCache**( )
java	void **clearCache**( )
py	**clearCache**( )
php	function **clearCache**( )
es	async **clearCache**( )

Invalide le cache des valeurs courantes du port de YoctoHub. Force le prochain appel à une méthode get_xxx() ou loadxxx() pour charger les les données depuis le module.

hubport→describe() YHubPort

Retourne un court texte décrivant de manière non-ambigüe l'instance du port de YoctoHub au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     string describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      describe( )
php     function describe( )
es      async describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

Retourne :
une chaîne de caractères décrivant le port de YoctoHub (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

hubport→get_advertisedValue() YHubPort
hubport→advertisedValue()

Retourne la valeur courante du port de YoctoHub (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     string get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
dnp     string get_advertisedValue( )
java    String get_advertisedValue( )
uwp     async Task<string> get_advertisedValue( )
py      get_advertisedValue( )
php     function get_advertisedValue( )
es      async get_advertisedValue( )
cmd     YHubPort target get_advertisedValue
```

Retourne :
une chaîne de caractères représentant la valeur courante du port de YoctoHub (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

hubport→get_baudRate() YHubPort
hubport→baudRate()

Retourne la vitesse de transfert utilisée par le port de YoctoHub, en kbps.

```
js      function get_baudRate( )
cpp     int get_baudRate( )
m       -(int) baudRate
pas     LongInt get_baudRate( ): LongInt
vb      function get_baudRate( ) As Integer
cs      int get_baudRate( )
dnp     int get_baudRate( )
java    int get_baudRate( )
uwp     async Task<int> get_baudRate( )
py      get_baudRate( )
php     function get_baudRate( )
es      async get_baudRate( )
cmd     YHubPort target get_baudRate
```

La valeur par défaut est 1000 kbps, une valeur inférieure révèle des problèmes de communication.

Retourne :
un entier représentant la vitesse de transfert utilisée par le port de YoctoHub, en kbps

En cas d'erreur, déclenche une exception ou retourne `Y_BAUDRATE_INVALID`.

hubport→get_enabled()
hubport→enabled()

YHubPort

Retourne vrai si le port du YoctoHub est alimenté, faux sinon.

```
js      function get_enabled( )
cpp     Y_ENABLED_enum get_enabled( )
m       -(Y_ENABLED_enum) enabled
pas     Integer get_enabled( ): Integer
vb      function get_enabled( ) As Integer
cs      int get_enabled( )
dnp     int get_enabled( )
java    int get_enabled( )
uwp     async Task<int> get_enabled( )
py      get_enabled( )
php     function get_enabled( )
es      async get_enabled( )
cmd     YHubPort target get_enabled
```

Retourne :

soit Y_ENABLED_FALSE, soit Y_ENABLED_TRUE, selon vrai si le port du YoctoHub est alimenté, faux sinon

En cas d'erreur, déclenche une exception ou retourne Y_ENABLED_INVALID.

hubport→get_errorMessage() YHubPort
hubport→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du port de YoctoHub.

```
js      function get_errorMessage( )
cpp     string get_errorMessage( )
m       -(NSString*) errorMessage
pas     string get_errorMessage( ): string
vb      function get_errorMessage( ) As String
cs      string get_errorMessage( )
java    String get_errorMessage( )
py      get_errorMessage( )
php     function get_errorMessage( )
es      get_errorMessage( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du port de YoctoHub.

hubport→get_errorType() YHubPort
hubport→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du port de YoctoHub.

```
js      function get_errorType( )
cpp     YRETCODE get_errorType( )
m       -(YRETCODE) errorType
pas     YRETCODE get_errorType( ): YRETCODE
vb      function get_errorType( ) As YRETCODE
cs      YRETCODE get_errorType( )
java    int get_errorType( )
py      get_errorType( )
php     function get_errorType( )
es      get_errorType( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du port de YoctoHub.

hubport→get_friendlyName() YHubPort
hubport→friendlyName()

Retourne un identifiant global du port de YoctoHub au format NOM_MODULE.NOM_FONCTION.

js	function **get_friendlyName**( )
cpp	string **get_friendlyName**( )
m	-(NSString*) friendlyName
cs	string **get_friendlyName**( )
dnp	string **get_friendlyName**( )
java	String **get_friendlyName**( )
py	**get_friendlyName**( )
php	function **get_friendlyName**( )
es	async **get_friendlyName**( )

Le chaîne retournée utilise soit les noms logiques du module et du port de YoctoHub si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel du port de YoctoHub (par exemple: MyCustomName.relay1)

Retourne :

une chaîne de caractères identifiant le port de YoctoHub en utilisant les noms logiques (ex: MyCustomName.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

hubport→get_functionDescriptor() YHubPort
hubport→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

```
js      function get_functionDescriptor( )
cpp     YFUN_DESCR get_functionDescriptor( )
m       -(YFUN_DESCR) functionDescriptor
pas     YFUN_DESCR get_functionDescriptor( ): YFUN_DESCR
vb      function get_functionDescriptor( ) As YFUN_DESCR
cs      YFUN_DESCR get_functionDescriptor( )
java    String get_functionDescriptor( )
py      get_functionDescriptor( )
php     function get_functionDescriptor( )
es      async get_functionDescriptor( )
```

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

Retourne :
un identifiant de type YFUN_DESCR.

Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

hubport→get_functionId() YHubPort
hubport→functionId()

Retourne l'identifiant matériel du port de YoctoHub, sans référence au module.

js	function **get_functionId**( )
cpp	string **get_functionId**( )
m	-(NSString*) functionId
vb	function **get_functionId**( ) As String
cs	string **get_functionId**( )
dnp	string **get_functionId**( )
java	String **get_functionId**( )
py	**get_functionId**( )
php	function **get_functionId**( )
es	async **get_functionId**( )

Par example `relay1`.

Retourne :
une chaîne de caractères identifiant le port de YoctoHub (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

hubport→get_hardwareId()
hubport→hardwareId()

YHubPort

Retourne l'identifiant matériel unique du port de YoctoHub au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
dnp     string get_hardwareId( )
java    String get_hardwareId( )
py      get_hardwareId( )
php     function get_hardwareId( )
es      async get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du port de YoctoHub (par example RELAYLO1-123456.relay1).

Retourne :
une chaîne de caractères identifiant le port de YoctoHub (ex: RELAYLO1-123456.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

hubport→get_logicalName() YHubPort
hubport→logicalName()

Retourne le nom logique du port de YoctoHub.

```
js      function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     string get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
dnp     string get_logicalName( )
java    String get_logicalName( )
uwp     async Task<string> get_logicalName( )
py      get_logicalName( )
php     function get_logicalName( )
es      async get_logicalName( )
cmd     YHubPort target get_logicalName
```

Retourne :
une chaîne de caractères représentant le nom logique du port de YoctoHub.

En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

hubport→get_module()
hubport→module()

YHubPort

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function **get_module**( )
cpp	YModule * **get_module**( )
m	-(YModule*) module
pas	TYModule **get_module**( ): TYModule
vb	function **get_module**( ) As YModule
cs	YModule **get_module**( )
dnp	YModuleProxy **get_module**( )
java	YModule **get_module**( )
py	**get_module**( )
php	function **get_module**( )
es	async **get_module**( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Retourne :
une instance de YModule

hubport→get_module_async() YHubPort
hubport→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

hubport→get_portState()
hubport→portState()

YHubPort

Retourne l'état actuel du port de YoctoHub.

```
js      function get_portState( )
cpp     Y_PORTSTATE_enum get_portState( )
m       -(Y_PORTSTATE_enum) portState
pas     Integer get_portState( ): Integer
vb      function get_portState( ) As Integer
cs      int get_portState( )
dnp     int get_portState( )
java    int get_portState( )
uwp     async Task<int> get_portState( )
py      get_portState( )
php     function get_portState( )
es      async get_portState( )
cmd     YHubPort target get_portState
```

Retourne :

une valeur parmi `Y_PORTSTATE_OFF`, `Y_PORTSTATE_OVRLD`, `Y_PORTSTATE_ON`, `Y_PORTSTATE_RUN` et `Y_PORTSTATE_PROG` représentant l'état actuel du port de YoctoHub

En cas d'erreur, déclenche une exception ou retourne `Y_PORTSTATE_INVALID`.

hubport→get_serialNumber() YHubPort
hubport→serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

```
js      function get_serialNumber( )
cpp     string get_serialNumber( )
m       -(NSString*) serialNumber
pas     string get_serialNumber( ): string
vb      function get_serialNumber( ) As String
cs      string get_serialNumber( )
dnp     string get_serialNumber( )
java    String get_serialNumber( )
uwp     async Task<string> get_serialNumber( )
py      get_serialNumber( )
php     function get_serialNumber( )
es      async get_serialNumber( )
cmd     YHubPort target get_serialNumber
```

Retourne :

: une chaîne de caractères représentant le numéro de série du module, préprogrammé en usine.

En cas d'erreur, déclenche une exception ou retourne YModule.SERIALNUMBER_INVALID.

hubport→get_userData()
hubport→userData()

YHubPort

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

```
js      function get_userData( )
cpp     void * get_userData( )
m       -(id) userData
pas     Tobject get_userData( ): Tobject
vb      function get_userData( ) As Object
cs      object get_userData( )
java    Object get_userData( )
py      get_userData( )
php     function get_userData( )
es      async get_userData( )
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Retourne :
l'objet stocké précédemment par l'appelant.

hubport→isOnline() YHubPort

Vérifie si le module hébergeant le port de YoctoHub est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     boolean isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
dnp     bool isOnline( )
java    boolean isOnline( )
py      isOnline( )
php     function isOnline( )
es      async isOnline( )
```

Si les valeurs des attributs en cache du port de YoctoHub sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Retourne :
`true` si le port de YoctoHub est joignable, `false` sinon

hubport→isOnline_async() YHubPort

Vérifie si le module hébergeant le port de YoctoHub est joignable, sans déclencher d'erreur.

```
js  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du port de YoctoHub sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

hubport→isReadOnly() YHubPort

Test si la fonction est en lecture seule.

cpp	bool **isReadOnly**( )
m	-(bool) **isReadOnly**
pas	boolean **isReadOnly**( ): boolean
vb	function **isReadOnly**( ) As Boolean
cs	bool **isReadOnly**( )
dnp	bool **isReadOnly**( )
java	boolean **isReadOnly**( )
uwp	async Task<bool> **isReadOnly**( )
py	**isReadOnly**( )
php	function **isReadOnly**( )
es	async **isReadOnly**( )
cmd	YHubPort **target** **isReadOnly**

Retourne vrais si la fonction est protégé en ecriture ou que la fontion n'est pas disponible.

Retourne :
`true` si la fonction est protégé en ecriture ou que la fontion n'est pas disponible

hubport→load() YHubPort

Met en cache les valeurs courantes du port de YoctoHub, avec une durée de validité spécifiée.

```
js      function load( msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (u64) msValidity
pas     YRETCODE load( msValidity: u64): YRETCODE
vb      function load( ByVal msValidity As Long) As YRETCODE
cs      YRETCODE load( ulong msValidity)
java    int load( long msValidity)
py      load( msValidity)
php     function load( $msValidity)
es      async load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

hubport→loadAttribute() YHubPort

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

```
js      function loadAttribute( attrName)
cpp     string loadAttribute( string attrName)
m       -(NSString*) loadAttribute : (NSString*) attrName
pas     string loadAttribute( attrName: string): string
vb      function loadAttribute( ) As String
cs      string loadAttribute( string attrName)
dnp     string loadAttribute( string attrName)
java    String loadAttribute( String attrName)
uwp     async Task<string> loadAttribute( string attrName)
py      loadAttribute( attrName)
php     function loadAttribute( $attrName)
es      async loadAttribute( attrName)
```

Paramètres :

attrName le nom de l'attribut désiré

Retourne :

une chaîne de caractères représentant la valeur actuelle de l'attribut.

En cas d'erreur, déclenche une exception ou retourne un chaîne vide.

hubport→load_async() YHubPort

Met en cache les valeurs courantes du port de YoctoHub, avec une durée de validité spécifiée.

```
js  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

msValidity	un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
callback	fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)
context	contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

hubport→muteValueCallbacks() YHubPort

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

js	function **muteValueCallbacks**( )
cpp	int **muteValueCallbacks**( )
m	-(int) **muteValueCallbacks**
pas	LongInt **muteValueCallbacks**( ): LongInt
vb	function **muteValueCallbacks**( ) As Integer
cs	int **muteValueCallbacks**( )
dnp	int **muteValueCallbacks**( )
java	int **muteValueCallbacks**( )
uwp	async Task<int> **muteValueCallbacks**( )
py	**muteValueCallbacks**( )
php	function **muteValueCallbacks**( )
es	async **muteValueCallbacks**( )
cmd	YHubPort **target** **muteValueCallbacks**

Vous pouvez utiliser cette fonction pour économiser la bande passante et le CPU sur les machines de faible puissance, ou pour éviter le déclanchement de callbacks HTTP. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

hubport→nextHubPort() YHubPort

Continue l'énumération des ports de YoctoHub commencée à l'aide de `yFirstHubPort()`. Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les ports de YoctoHub sont retournés.

```
js      function nextHubPort( )
cpp     YHubPort * nextHubPort( )
m       -(YHubPort*) nextHubPort
pas     TYHubPort nextHubPort( ): TYHubPort
vb      function nextHubPort( ) As YHubPort
cs      YHubPort nextHubPort( )
java    YHubPort nextHubPort( )
uwp     YHubPort nextHubPort( )
py      nextHubPort( )
php     function nextHubPort( )
es      nextHubPort( )
```

Si vous souhaitez retrouver un port de YoctoHub spécifique, utilisez `HubPort.findHubPort()` avec un hardwareID ou un nom logique.

Retourne :
un pointeur sur un objet `YHubPort` accessible en ligne, ou `null` lorsque l'énumération est terminée.

hubport→registerValueCallback() YHubPort

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function **registerValueCallback**(**callback**)
cpp	int **registerValueCallback**(YHubPortValueCallback **callback**)
m	-(int) **registerValueCallback** : (YHubPortValueCallback) **callback**
pas	LongInt **registerValueCallback**(**callback**: TYHubPortValueCallback): LongInt
vb	function **registerValueCallback**( ) As Integer
cs	int **registerValueCallback**(ValueCallback **callback**)
java	int **registerValueCallback**(UpdateCallback **callback**)
uwp	async Task<int> **registerValueCallback**(ValueCallback **callback**)
py	**registerValueCallback**(**callback**)
php	function **registerValueCallback**(**$callback**)
es	async **registerValueCallback**(**callback**)

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

Paramètres :

callback la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

hubport→set_enabled() YHubPort
hubport→setEnabled()

Modifie le mode d'activation du port du YoctoHub.

js	function **set_enabled**(**newval**)
cpp	int **set_enabled**(Y_ENABLED_enum **newval**)
m	-(int) setEnabled : (Y_ENABLED_enum) **newval**
pas	integer **set_enabled**(**newval**: Integer): integer
vb	function **set_enabled**(ByVal **newval** As Integer) As Integer
cs	int **set_enabled**(int **newval**)
dnp	int **set_enabled**(int **newval**)
java	int **set_enabled**(int **newval**)
uwp	async Task<int> **set_enabled**(int **newval**)
py	**set_enabled**(**newval**)
php	function **set_enabled**(**$newval**)
es	async **set_enabled**(**newval**)
cmd	YHubPort **target set_enabled newval**

Si le port est actif, il sera alimenté. Sinon, l'alimentation du module est coupée.

Paramètres :

newval soit Y_ENABLED_FALSE, soit Y_ENABLED_TRUE, selon le mode d'activation du port du YoctoHub

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

hubport→set_logicalName() YHubPort
hubport→setLogicalName()

Modifie le nom logique du port de YoctoHub.

js	function **set_logicalName**(**newval**)
cpp	int **set_logicalName**(string **newval**)
m	-(int) setLogicalName : (NSString*) **newval**
pas	integer **set_logicalName**(**newval**: string): integer
vb	function **set_logicalName**(ByVal **newval** As String) As Integer
cs	int **set_logicalName**(string **newval**)
dnp	int **set_logicalName**(string **newval**)
java	int **set_logicalName**(String **newval**)
uwp	async Task<int> **set_logicalName**(string **newval**)
py	**set_logicalName**(**newval**)
php	function **set_logicalName**(**$newval**)
es	async **set_logicalName**(**newval**)
cmd	YHubPort **target set_logicalName newval**

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom logique du port de YoctoHub.

Retourne :

`YAPI_SUCCESS` si l'appel se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

hubport→set_userData() YHubPort
hubport→setUserData()

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

js	function **set_userData**(**data**)
cpp	void **set_userData**(void * **data**)
m	-(void) setUserData : (id) **data**
pas	**set_userData**(**data**: Tobject)
vb	procedure **set_userData**(ByVal **data** As Object)
cs	void **set_userData**(object **data**)
java	void **set_userData**(Object **data**)
py	**set_userData**(**data**)
php	function **set_userData**(**$data**)
es	async **set_userData**(**data**)

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Paramètres :

data objet quelconque à mémoriser

hubport→unmuteValueCallbacks() YHubPort

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

js	function **unmuteValueCallbacks**( )
cpp	int **unmuteValueCallbacks**( )
m	-(int) **unmuteValueCallbacks**
pas	LongInt **unmuteValueCallbacks**( ): LongInt
vb	function **unmuteValueCallbacks**( ) As Integer
cs	int **unmuteValueCallbacks**( )
dnp	int **unmuteValueCallbacks**( )
java	int **unmuteValueCallbacks**( )
uwp	async Task<int> **unmuteValueCallbacks**( )
py	**unmuteValueCallbacks**( )
php	function **unmuteValueCallbacks**( )
es	async **unmuteValueCallbacks**( )
cmd	YHubPort **target** **unmuteValueCallbacks**

Cette fonction annule un précédent appel à `muteValueCallbacks()`. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

hubport→wait_async() YHubPort

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js  function wait_async( callback, context)
es  wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

Paramètres :

callback fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout.

8.2. La classe YCellular

Interface pour intéragir avec les interfaces réseau cellulaire, disponibles par exemple dans le YoctoHub-GSM-2G, le YoctoHub-GSM-3G-EU et le YoctoHub-GSM-3G-NA

La classe `YCellular` permet de configurer et de contrôler la configuration du réseau cellulaire sur les modules Yoctopuce qui en sont dotés. Notez que les paramètres TCP/IP sont configurés séparément, à l'aide de la classe `YNetwork`.

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_cellular.js'></script>
cpp	#include "yocto_cellular.h"
m	#import "yocto_cellular.h"
pas	uses yocto_cellular;
vb	yocto_cellular.vb
cs	yocto_cellular.cs
dnp	import YoctoProxyAPI.YCellularProxy
java	import com.yoctopuce.YoctoAPI.YCellular;
uwp	import com.yoctopuce.YoctoAPI.YCellular;
py	from yocto_cellular import *
php	require_once('yocto_cellular.php');
es	in HTML: <script src="../../lib/yocto_cellular.js"></script> in node.js: require('yoctolib-es2017/yocto_cellular.js');
vi	YCellular.vi

Fonction globales

YCellular.FindCellular(**func**)

Permet de retrouver une interface cellulaire d'après un identifiant donné.

YCellular.FindCellularInContext(**yctx**, **func**)

Permet de retrouver une interface cellulaire d'après un identifiant donné dans un Context YAPI.

YCellular.FirstCellular()

Commence l'énumération des interfaces réseau cellulaire accessibles par la librairie.

YCellular.FirstCellularInContext(**yctx**)

Commence l'énumération des interfaces réseau cellulaire accessibles par la librairie.

YCellular.GetSimilarFunctions()

Enumère toutes les fonctions de type Cellular disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).

Propriétés des objets `YCellularProxy`

cellular→AdvertisedValue *[lecture seule]*

Courte chaîne de caractères représentant l'état courant de la fonction.

cellular→Apn *[modifiable]*

Nom du point d'accès (APN) à utiliser, si nécessaire.

cellular→EnableData *[modifiable]*

Condition dans laquelle le service de données IP (GRPS) doit être activé.

cellular→FriendlyName *[lecture seule]*

Identifiant global de la fonction au format `NOM_MODULE.NOM_FONCTION`.

cellular→FunctionId *[lecture seule]*

Identifiant matériel de l'interface cellulaire, sans référence au module.

cellular→HardwareId *[lecture seule]*

Identifiant matériel unique de la fonction au format SERIAL.FUNCTIONID.

cellular→IsOnline *[lecture seule]*

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

cellular→LockedOperator *[modifiable]*

Nom de l'opérateur de réseau cellulaire à utiliser exclusivement, si le choix automatique est désactivé, ou une chaîne vide si la carte SIM sélectionne automatiquement l'opérateur selon ceux disponibles.

cellular→LogicalName *[modifiable]*

Nom logique de la fonction.

cellular→Pin *[modifiable]*

String opaque si un code PIN a été configuré dans le module pour accéder à la carte SIM, ou une chaîne vide il n'a pas été configuré ou si la SIM a rejeté le code indiqué.

cellular→PingInterval *[modifiable]*

Intervalle entre les tests de connectivité spontanés, en secondes.

cellular→SerialNumber *[lecture seule]*

Numéro de série du module, préprogrammé en usine.

Méthodes des objets YCellular

cellular→_AT(cmd)

Envoie une commande AT au module GSM, et retourne le résultat.

cellular→clearCache()

Invalide le cache.

cellular→clearDataCounters()

Réinitialise les compteurs de données transmises et reçues.

cellular→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'interface cellulaire au format TYPE(NAME)=SERIAL.FUNCTIONID.

cellular→get_advertisedValue()

Retourne la valeur courante de l'interface cellulaire (pas plus de 6 caractères).

cellular→get_airplaneMode()

Retourne vrai si le mode avion est activé (radio désactivée).

cellular→get_apn()

Retourne le nom du point d'accès (APN) à utiliser, si nécessaire.

cellular→get_apnSecret()

Retourne une string opaque si des paramètres d'identification sur l'APN ont été configurés dans le module, ou une chaîne vide autrement.

cellular→get_availableOperators()

Retourne la liste des opérateurs GSM disponibles à proximité.

cellular→get_cellIdentifier()

Retourne l'identifiant unique de la station de base utilisée: MCC, MNC, LAC et Cell ID.

cellular→get_cellOperator()

Retourne le nom de l'opérateur de réseau cellulaire actuellement utilisé.

cellular→get_cellType()

Type de connection cellulaire active.

cellular→get_dataReceived()

Retourne le nombre d'octets reçus jusqu'à présent.

cellular→get_dataSent()

Retourne le nombre d'octets envoyés jusqu'à présent.

cellular→get_enableData()

Retourne la condition dans laquelle le service de données IP (GRPS) doit être activé.

cellular→get_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface cellulaire.

cellular→get_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface cellulaire.

cellular→get_friendlyName()

Retourne un identifiant global de l'interface cellulaire au format `NOM_MODULE.NOM_FONCTION`.

cellular→get_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

cellular→get_functionId()

Retourne l'identifiant matériel de l'interface cellulaire, sans référence au module.

cellular→get_hardwareId()

Retourne l'identifiant matériel unique de l'interface cellulaire au format `SERIAL.FUNCTIONID`.

cellular→get_imsi()

Retourne une string opaque si un code PIN a été configuré dans le module pour accéder à la carte SIM, ou une chaîne vide il n'a pas été configuré ou si la SIM a rejeté le code indiqué.

cellular→get_linkQuality()

Retourne la qualité de la connection, exprimée en pourcents.

cellular→get_lockedOperator()

Retourne le nom de l'opérateur de réseau cellulaire à utiliser exclusivement, si le choix automatique est désactivé, ou une chaîne vide si la carte SIM sélectionne automatiquement l'opérateur selon ceux disponibles.

cellular→get_logicalName()

Retourne le nom logique de l'interface cellulaire.

cellular→get_message()

Retourne le dernier message de diagnostique de l'interface au réseau sans fil.

cellular→get_module()

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

cellular→get_module_async(**callback**, **context**)

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

cellular→get_pin()

Retourne une string opaque si un code PIN a été configuré dans le module pour accéder à la carte SIM, ou une chaîne vide il n'a pas été configuré ou si la SIM a rejeté le code indiqué.

cellular→get_pingInterval()

Retourne l'intervalle entre les tests de connectivité spontanés, en secondes.

cellular→get_serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

cellular→get_userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

cellular→isOnline()

Vérifie si le module hébergeant l'interface cellulaire est joignable, sans déclencher d'erreur.

cellular→isOnline_async(**callback**, **context**)

Vérifie si le module hébergeant l'interface cellulaire est joignable, sans déclencher d'erreur.

cellular→isReadOnly()

Test si la fonction est en lecture seule.

cellular→load(msValidity)

Met en cache les valeurs courantes de l'interface cellulaire, avec une durée de validité spécifiée.

cellular→loadAttribute(attrName)

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

cellular→load_async(msValidity, callback, context)

Met en cache les valeurs courantes de l'interface cellulaire, avec une durée de validité spécifiée.

cellular→muteValueCallbacks()

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

cellular→nextCellular()

Continue l'énumération des interfaces réseau cellulaire commencée à l'aide de `yFirstCellular()`
Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les interfaces réseau cellulaire sont retournés.

cellular→quickCellSurvey()

Retourne la liste d'identifiants pour les antennes GSM à proximité, telle que requise pour géolocaliser rapidement le module.

cellular→registerValueCallback(callback)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

cellular→sendPUK(puk, newPin)

Envoie le code PUK à la carte SIM pour la débloquer après trois échecs consécutifs de code PIN, et établit un nouveau code PIN dans la SIM.

cellular→set_airplaneMode(newval)

Modifie l'état du mode avion (radio désactivée).

cellular→set_apn(newval)

Retourne le nom du point d'accès (APN) à utiliser, si nécessaire.

cellular→set_apnAuth(username, password)

Configure les paramètres d'identification pour se connecter à l'APN.

cellular→set_dataReceived(newval)

Modifie la valeur du compteur d'octets reçus.

cellular→set_dataSent(newval)

Modifie la valeur du compteur d'octets envoyés.

cellular→set_enableData(newval)

Modifie la condition dans laquelle le service de données IP (GRPS) doit être activé.

cellular→set_lockedOperator(newval)

Modifie le nom de l'opérateur de réseau cellulaire à utiliser.

cellular→set_logicalName(newval)

Modifie le nom logique de l'interface cellulaire.

cellular→set_pin(newval)

Modifie le code PIN utilisé par le module pour accéder à la carte SIM.

cellular→set_pingInterval(newval)

Modifie l'intervalle entre les tests de connectivité spontanés, en secondes.

cellular→set_userData(data)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

cellular→unmuteValueCallbacks()

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

cellular→wait_async(callback, context)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

YCellular.FindCellular() YCellular
YCellular.FindCellular()

Permet de retrouver une interface cellulaire d'après un identifiant donné.

```
js      function yFindCellular( func)
cpp     YCellular* yFindCellular( string func)
m       +(YCellular*) FindCellular : (NSString*) func
pas     TYCellular yFindCellular( func: string): TYCellular
vb      function yFindCellular( ByVal func As String) As YCellular
cs      static YCellular FindCellular( string func)
dnp     static YCellularProxy FindCellular( string func)
java    static YCellular FindCellular( String func)
uwp     static YCellular FindCellular( string func)
py      FindCellular( func)
php     function yFindCellular( $func)
es      static FindCellular( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'interface cellulaire soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YCellular.isOnline()` pour tester si l'interface cellulaire est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Si un appel à la méthode is_online() de cet objet renvoie FAUX alors que vous êtes sûr que le module correspondant est bien branché, vérifiez que vous n'avez pas oublié d'appeler registerHub() à l'initialisation de de l'application.

Paramètres :

func une chaîne de caractères qui référence l'interface cellulaire sans ambiguïté, par exemple `YHUBGSM1.cellular`.

Retourne :

un objet de classe `YCellular` qui permet ensuite de contrôler l'interface cellulaire.

YCellular.FindCellularInContext() YCellular
YCellular.FindCellularInContext()

Permet de retrouver une interface cellulaire d'après un identifiant donné dans un Context YAPI.

java static YCellular **FindCellularInContext**(YAPIContext **yctx**, String **func**)
uwp static YCellular **FindCellularInContext**(YAPIContext **yctx**, string **func**)
es static **FindCellularInContext**(**yctx**, **func**)

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'interface cellulaire soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YCellular.isOnline()` pour tester si l'interface cellulaire est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Paramètres :

yctx un contexte YAPI

func une chaîne de caractères qui référence l'interface cellulaire sans ambiguïté, par exemple `YHUBGSM1.cellular`.

Retourne :

un objet de classe `YCellular` qui permet ensuite de contrôler l'interface cellulaire.

YCellular.FirstCellular() YCellular
YCellular.FirstCellular()

Commence l'énumération des interfaces réseau cellulaire accessibles par la librairie.

js	function **yFirstCellular**( )
cpp	YCellular * **yFirstCellular**( )
m	+(YCellular*) **FirstCellular**
pas	TYCellular **yFirstCellular**( ): TYCellular
vb	function **yFirstCellular**( ) As YCellular
cs	static YCellular **FirstCellular**( )
java	static YCellular **FirstCellular**( )
uwp	static YCellular **FirstCellular**( )
py	**FirstCellular**( )
php	function **yFirstCellular**( )
es	static **FirstCellular**( )

Utiliser la fonction `YCellular.nextCellular()` pour itérer sur les autres interfaces réseau cellulaire.

Retourne :

un pointeur sur un objet `YCellular`, correspondant à la première interface cellulaire accessible en ligne, ou `null` si il n'y a pas de interfaces réseau cellulaire disponibles.

YCellular.FirstCellularInContext() YCellular
YCellular.FirstCellularInContext()

Commence l'énumération des interfaces réseau cellulaire accessibles par la librairie.

`java` static YCellular **FirstCellularInContext**(YAPIContext **yctx**)
`uwp` static YCellular **FirstCellularInContext**(YAPIContext **yctx**)
`es` static **FirstCellularInContext**(**yctx**)

Utiliser la fonction `YCellular.nextCellular()` pour itérer sur les autres interfaces réseau cellulaire.

Paramètres :

yctx un contexte YAPI.

Retourne :

un pointeur sur un objet `YCellular`, correspondant à la première interface cellulaire accessible en ligne, ou `null` si il n'y a pas de interfaces réseau cellulaire disponibles.

YCellular.GetSimilarFunctions() YCellular
YCellular.GetSimilarFunctions()

Enumère toutes les fonctions de type Cellular disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).

dnp static new string[] **GetSimilarFunctions**( )

Chaque chaîne retournée peut être passée en argument à la méthode `YCellular.FindCellular` pour obtenir une objet permettant d'intéragir avec le module correspondant.

Retourne :

un tableau de chaînes de caractères, contenant les identifiants matériels de chaque fonction disponible trouvée.

cellular→AdvertisedValue — YCellular

Courte chaîne de caractères représentant l'état courant de la fonction.

dnp string **AdvertisedValue**

cellular→Apn YCellular

Nom du point d'accès (APN) à utiliser, si nécessaire.

dnp string **Apn**

Lorsque l'APN est vide, celui proposé par l'opérateur cellulaire est utilisée.

Modifiable. Retourne le nom du point d'accès (APN) à utiliser, si nécessaire. Lorsque l'APN est vide, celui proposé par l'opérateur cellulaire est utilisée. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

cellular→EnableData — YCellular

Condition dans laquelle le service de données IP (GRPS) doit être activé.

dnp int **EnableData**

Lorsque le service de donnée n'est pas actif, seules les communications par SMS sont possibles.

Valeurs possibles:

Y_ENABLEDATA_INVALID	= 0
Y_ENABLEDATA_HOMENETWORK	= 1
Y_ENABLEDATA_ROAMING	= 2
Y_ENABLEDATA_NEVER	= 3
Y_ENABLEDATA_NEUTRALITY	= 4

Modifiable. Le service peut être soit complètement désactivé, soit limité au réseau de de l'émetteur de la carte SIM, soit être activé pour tous les réseaux en partenariat avec la carte SIM (roaming). Attention, l'utilisation de données en roaming peut conduire à des coûts de télécommunication exhorbitants !
Lorsque le service de donnée n'est pas actif, seules les communications par SMS sont possibles. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

cellular→FriendlyName YCellular

Identifiant global de la fonction au format NOM_MODULE.NOM_FONCTION.

dnp string **FriendlyName**

Le chaîne retournée utilise soit les noms logiques du module et de la fonction si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel de la fonction (par exemple: MyCustomName.relay1)

cellular→FunctionId YCellular

Identifiant matériel de l'interface cellulaire, sans référence au module.

dnp string **FunctionId**

Par example `relay1`.

cellular→HardwareId — YCellular

Identifiant matériel unique de la fonction au format SERIAL.FUNCTIONID.

dnp string **HardwareId**

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de la fonction (par example RELAYLO1-123456.relay1).

cellular→IsOnline YCellular

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

dnp bool **IsOnline**

Si les valeurs des attributs en cache de la fonction sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

cellular→LockedOperator YCellular

Nom de l'opérateur de réseau cellulaire à utiliser exclusivement, si le choix automatique est désactivé, ou une chaîne vide si la carte SIM sélectionne automatiquement l'opérateur selon ceux disponibles.

dnp string **LockedOperator**

Modifiable. Modifie le nom de l'opérateur de réseau cellulaire à utiliser. Si le nom est une chaîne vide, le choix sera fait automatiquement selon la carte SIM. Sinon, seul l'opérateur choisi sera utilisé. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

cellular→LogicalName YCellular

Nom logique de la fonction.

dnp string **LogicalName**

Modifiable. Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

cellular→Pin YCellular

String opaque si un code PIN a été configuré dans le module pour accéder à la carte SIM, ou une chaîne vide il n'a pas été configuré ou si la SIM a rejeté le code indiqué.

dnp string **Pin**

Modifiable. Modifie le code PIN utilisé par le module pour accéder à la carte SIM. Cette fonction ne change pas le code sur la carte SIM elle-même, mais uniquement le paramètre utilisé par le module pour essayer d'en obtenir l'accès. Si le code SIM ne fonctionne pas dès le premier essai, il sera automatiquement oublié et un message "Enter SIM PIN" apparaître dans l'attribut 'message'. Il faudra alors appeler à nouveau cette méthode avec le bon code PIN. Après trois essais infructueux consécutifs le message devient "Enter SIM PUK" et il faut alors entrer le code PUK de la carte SIM avec la méthode `sendPUK`.

N'oubliez pas d'appeler la méthode `saveToFlash()` du module pour que le paramètre soit sauvegardé dans la flash.

cellular→PingInterval YCellular

Intervalle entre les tests de connectivité spontanés, en secondes.

dnp int **PingInterval**

Modifiable. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

cellular→SerialNumber — YCellular

Numéro de série du module, préprogrammé en usine.

dnp string **SerialNumber**

cellular→_AT() YCellular

Envoie une commande AT au module GSM, et retourne le résultat.

```
cpp     string _AT( string cmd)
m       -(NSString*) _AT : (NSString*) cmd
pas     string _AT( cmd: string): string
vb      function _AT( ) As String
cs      string _AT( string cmd)
dnp     string _AT( string cmd)
java    String _AT( String cmd)
uwp     async Task<string> _AT( string cmd)
py      _AT( cmd)
php     function _AT( $cmd)
es      async _AT( cmd)
cmd     YCellular target _AT cmd
```

La commande ne s'exécute que lorsque le module GSM dans un état standard, et doit le laisser exactement dans le même état. N'utilisez cette fonction qu'avec la plus grande prudence !

Paramètres :

cmd la commande AT à exécuter, comme par exemple: "+CCLK?"

Retourne :

une chaîne de caractères contenant le résultat de la commande. Les lignes vides sont automatiquement filtrées du résultat.

cellular→clearCache() YCellular

Invalide le cache.

js	function **clearCache**( )
cpp	void **clearCache**( )
m	-(void) **clearCache**
pas	**clearCache**( )
vb	procedure **clearCache**( )
cs	void **clearCache**( )
java	void **clearCache**( )
py	**clearCache**( )
php	function **clearCache**( )
es	async **clearCache**( )

Invalide le cache des valeurs courantes de l'interface cellulaire. Force le prochain appel à une méthode get_xxx() ou loadxxx() pour charger les les données depuis le module.

cellular→clearDataCounters() YCellular

Réinitialise les compteurs de données transmises et reçues.

```
js      function clearDataCounters( )
cpp     int clearDataCounters( )
m       -(int) clearDataCounters
pas     LongInt clearDataCounters( ): LongInt
vb      function clearDataCounters( ) As Integer
cs      int clearDataCounters( )
dnp     int clearDataCounters( )
java    int clearDataCounters( )
uwp     async Task<int> clearDataCounters( )
py      clearDataCounters( )
php     function clearDataCounters( )
es      async clearDataCounters( )
cmd     YCellular target clearDataCounters
```

Retourne :
YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→describe() YCellular

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'interface cellulaire au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     string describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      describe( )
php     function describe( )
es      async describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

Retourne :
une chaîne de caractères décrivant l'interface cellulaire (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

cellular→get_advertisedValue() YCellular
cellular→advertisedValue()

Retourne la valeur courante de l'interface cellulaire (pas plus de 6 caractères).

js	function **get_advertisedValue**( )
cpp	string **get_advertisedValue**( )
m	-(NSString*) advertisedValue
pas	string **get_advertisedValue**( ): string
vb	function **get_advertisedValue**( ) As String
cs	string **get_advertisedValue**( )
dnp	string **get_advertisedValue**( )
java	String **get_advertisedValue**( )
uwp	async Task<string> **get_advertisedValue**( )
py	**get_advertisedValue**( )
php	function **get_advertisedValue**( )
es	async **get_advertisedValue**( )
cmd	YCellular **target get_advertisedValue**

Retourne :
une chaîne de caractères représentant la valeur courante de l'interface cellulaire (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

cellular→get_airplaneMode()
cellular→airplaneMode()

YCellular

Retourne vrai si le mode avion est activé (radio désactivée).

js	function **get_airplaneMode**( )
cpp	Y_AIRPLANEMODE_enum **get_airplaneMode**( )
m	-(Y_AIRPLANEMODE_enum) airplaneMode
pas	Integer **get_airplaneMode**( ): Integer
vb	function **get_airplaneMode**( ) As Integer
cs	int **get_airplaneMode**( )
dnp	int **get_airplaneMode**( )
java	int **get_airplaneMode**( )
uwp	async Task<int> **get_airplaneMode**( )
py	**get_airplaneMode**( )
php	function **get_airplaneMode**( )
es	async **get_airplaneMode**( )
cmd	YCellular **target get_airplaneMode**

Retourne :
soit `Y_AIRPLANEMODE_OFF`, soit `Y_AIRPLANEMODE_ON`, selon vrai si le mode avion est activé (radio désactivée)

En cas d'erreur, déclenche une exception ou retourne `Y_AIRPLANEMODE_INVALID`.

cellular→get_apn()
cellular→apn()

YCellular

Retourne le nom du point d'accès (APN) à utiliser, si nécessaire.

```
js    function get_apn( )
cpp   string get_apn( )
m     -(NSString*) apn
pas   string get_apn( ): string
vb    function get_apn( ) As String
cs    string get_apn( )
dnp   string get_apn( )
java  String get_apn( )
uwp   async Task<string> get_apn( )
py    get_apn( )
php   function get_apn( )
es    async get_apn( )
cmd   YCellular target get_apn
```

Lorsque l'APN est vide, celui proposé par l'opérateur cellulaire est utilisée.

Retourne :
une chaîne de caractères représentant le nom du point d'accès (APN) à utiliser, si nécessaire

En cas d'erreur, déclenche une exception ou retourne `Y_APN_INVALID`.

cellular→get_apnSecret()
cellular→apnSecret()

YCellular

Retourne une string opaque si des paramètres d'identification sur l'APN ont été configurés dans le module, ou une chaîne vide autrement.

```
js      function get_apnSecret( )
cpp     string get_apnSecret( )
m       -(NSString*) apnSecret
pas     string get_apnSecret( ): string
vb      function get_apnSecret( ) As String
cs      string get_apnSecret( )
dnp     string get_apnSecret( )
java    String get_apnSecret( )
uwp     async Task<string> get_apnSecret( )
py      get_apnSecret( )
php     function get_apnSecret( )
es      async get_apnSecret( )
cmd     YCellular target get_apnSecret
```

Pour configurer ces paramètres, utilisez la méthode `set_apnAuth()`.

Retourne :

une chaîne de caractères représentant une string opaque si des paramètres d'identification sur l'APN ont été configurés dans le module, ou une chaîne vide autrement

En cas d'erreur, déclenche une exception ou retourne `Y_APNSECRET_INVALID`.

cellular→get_availableOperators() YCellular
cellular→availableOperators()

Retourne la liste des opérateurs GSM disponibles à proximité.

```
js    function get_availableOperators( )
cpp   vector<string> get_availableOperators( )
m     -(NSMutableArray*) availableOperators
pas   TStringArray get_availableOperators( ): TStringArray
vb    function get_availableOperators( ) As List
cs    List<string> get_availableOperators( )
dnp   string[] get_availableOperators( )
java  ArrayList<String> get_availableOperators( )
uwp   async Task<List<string>> get_availableOperators( )
py    get_availableOperators( )
php   function get_availableOperators( )
es    async get_availableOperators( )
cmd   YCellular target get_availableOperators
```

Cette fonction peut typiquement prendre 30 secondes à une minute pour rendre la main. Notez qu'en général une SIM ne permet de se connecter qu'à certains opérateur, et donc pas forcément à tous ceux listés par cette fonction.

Retourne :
une liste de noms d'opérateur.

cellular→get_cellIdentifier()
cellular→cellIdentifier()

YCellular

Retourne l'identifiant unique de la station de base utilisée: MCC, MNC, LAC et Cell ID.

```
js      function get_cellIdentifier( )
cpp     string get_cellIdentifier( )
m       -(NSString*) cellIdentifier
pas     string get_cellIdentifier( ): string
vb      function get_cellIdentifier( ) As String
cs      string get_cellIdentifier( )
dnp     string get_cellIdentifier( )
java    String get_cellIdentifier( )
uwp     async Task<string> get_cellIdentifier( )
py      get_cellIdentifier( )
php     function get_cellIdentifier( )
es      async get_cellIdentifier( )
cmd     YCellular target get_cellIdentifier
```

Retourne :

une chaîne de caractères représentant l'identifiant unique de la station de base utilisée: MCC, MNC, LAC et Cell ID

En cas d'erreur, déclenche une exception ou retourne `Y_CELLIDENTIFIER_INVALID`.

cellular→get_cellOperator() YCellular
cellular→cellOperator()

Retourne le nom de l'opérateur de réseau cellulaire actuellement utilisé.

js	function **get_cellOperator**( )
cpp	string **get_cellOperator**( )
m	-(NSString*) cellOperator
pas	string **get_cellOperator**( ): string
vb	function **get_cellOperator**( ) As String
cs	string **get_cellOperator**( )
dnp	string **get_cellOperator**( )
java	String **get_cellOperator**( )
uwp	async Task<string> **get_cellOperator**( )
py	**get_cellOperator**( )
php	function **get_cellOperator**( )
es	async **get_cellOperator**( )
cmd	YCellular **target get_cellOperator**

Retourne :
une chaîne de caractères représentant le nom de l'opérateur de réseau cellulaire actuellement utilisé

En cas d'erreur, déclenche une exception ou retourne `Y_CELLOPERATOR_INVALID`.

cellular→get_cellType()
cellular→cellType()

YCellular

Type de connection cellulaire active.

js	function **get_cellType**( )
cpp	Y_CELLTYPE_enum **get_cellType**( )
m	-(Y_CELLTYPE_enum) cellType
pas	Integer **get_cellType**( ): Integer
vb	function **get_cellType**( ) As Integer
cs	int **get_cellType**( )
dnp	int **get_cellType**( )
java	int **get_cellType**( )
uwp	async Task<int> **get_cellType**( )
py	**get_cellType**( )
php	function **get_cellType**( )
es	async **get_cellType**( )
cmd	YCellular **target get_cellType**

Retourne :
une valeur parmi `Y_CELLTYPE_GPRS`, `Y_CELLTYPE_EGPRS`, `Y_CELLTYPE_WCDMA`, `Y_CELLTYPE_HSDPA`, `Y_CELLTYPE_NONE` et `Y_CELLTYPE_CDMA`

En cas d'erreur, déclenche une exception ou retourne `Y_CELLTYPE_INVALID`.

cellular→get_dataReceived() YCellular
cellular→dataReceived()

Retourne le nombre d'octets reçus jusqu'à présent.

```
js    function get_dataReceived( )
cpp   int get_dataReceived( )
m     -(int) dataReceived
pas   LongInt get_dataReceived( ): LongInt
vb    function get_dataReceived( ) As Integer
cs    int get_dataReceived( )
dnp   int get_dataReceived( )
java  int get_dataReceived( )
uwp   async Task<int> get_dataReceived( )
py    get_dataReceived( )
php   function get_dataReceived( )
es    async get_dataReceived( )
cmd   YCellular target get_dataReceived
```

Retourne :
un entier représentant le nombre d'octets reçus jusqu'à présent

En cas d'erreur, déclenche une exception ou retourne `Y_DATARECEIVED_INVALID`.

cellular→get_dataSent()
cellular→dataSent()

YCellular

Retourne le nombre d'octets envoyés jusqu'à présent.

```
js    function get_dataSent( )
cpp   int get_dataSent( )
m     -(int) dataSent
pas   LongInt get_dataSent( ): LongInt
vb    function get_dataSent( ) As Integer
cs    int get_dataSent( )
dnp   int get_dataSent( )
java  int get_dataSent( )
uwp   async Task<int> get_dataSent( )
py    get_dataSent( )
php   function get_dataSent( )
es    async get_dataSent( )
cmd   YCellular target get_dataSent
```

Retourne :
un entier représentant le nombre d'octets envoyés jusqu'à présent

En cas d'erreur, déclenche une exception ou retourne `Y_DATASENT_INVALID`.

cellular→get_enableData() YCellular
cellular→enableData()

Retourne la condition dans laquelle le service de données IP (GRPS) doit être activé.

js	function **get_enableData**( )
cpp	Y_ENABLEDATA_enum **get_enableData**( )
m	-(Y_ENABLEDATA_enum) enableData
pas	Integer **get_enableData**( ): Integer
vb	function **get_enableData**( ) As Integer
cs	int **get_enableData**( )
dnp	int **get_enableData**( )
java	int **get_enableData**( )
uwp	async Task<int> **get_enableData**( )
py	**get_enableData**( )
php	function **get_enableData**( )
es	async **get_enableData**( )
cmd	YCellular **target get_enableData**

Lorsque le service de donnée n'est pas actif, seules les communications par SMS sont possibles.

Retourne :

une valeur parmi `Y_ENABLEDATA_HOMENETWORK`, `Y_ENABLEDATA_ROAMING`, `Y_ENABLEDATA_NEVER` et `Y_ENABLEDATA_NEUTRALITY` représentant la condition dans laquelle le service de données IP (GRPS) doit être activé

En cas d'erreur, déclenche une exception ou retourne `Y_ENABLEDATA_INVALID`.

cellular→get_errorMessage() YCellular
cellular→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface cellulaire.

```
js      function get_errorMessage( )
cpp     string get_errorMessage( )
m       -(NSString*) errorMessage
pas     string get_errorMessage( ): string
vb      function get_errorMessage( ) As String
cs      string get_errorMessage( )
java    String get_errorMessage( )
py      get_errorMessage( )
php     function get_errorMessage( )
es      get_errorMessage( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'interface cellulaire.

cellular→get_errorType() YCellular
cellular→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface cellulaire.

js	function **get_errorType**( )
cpp	YRETCODE **get_errorType**( )
m	-(YRETCODE) errorType
pas	YRETCODE **get_errorType**( ): YRETCODE
vb	function **get_errorType**( ) As YRETCODE
cs	YRETCODE **get_errorType**( )
java	int **get_errorType**( )
py	**get_errorType**( )
php	function **get_errorType**( )
es	**get_errorType**( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'interface cellulaire.

cellular→get_friendlyName() YCellular
cellular→friendlyName()

Retourne un identifiant global de l'interface cellulaire au format NOM_MODULE.NOM_FONCTION.

```
js      function get_friendlyName( )
cpp     string get_friendlyName( )
m       -(NSString*) friendlyName
cs      string get_friendlyName( )
dnp     string get_friendlyName( )
java    String get_friendlyName( )
py      get_friendlyName( )
php     function get_friendlyName( )
es      async get_friendlyName( )
```

Le chaîne retournée utilise soit les noms logiques du module et de l'interface cellulaire si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel de l'interface cellulaire (par exemple: MyCustomName.relay1)

Retourne :

une chaîne de caractères identifiant l'interface cellulaire en utilisant les noms logiques (ex: MyCustomName.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

cellular→get_functionDescriptor() YCellular
cellular→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function **get_functionDescriptor**( )
cpp	YFUN_DESCR **get_functionDescriptor**( )
m	-(YFUN_DESCR) functionDescriptor
pas	YFUN_DESCR **get_functionDescriptor**( ): YFUN_DESCR
vb	function **get_functionDescriptor**( ) As YFUN_DESCR
cs	YFUN_DESCR **get_functionDescriptor**( )
java	String **get_functionDescriptor**( )
py	**get_functionDescriptor**( )
php	function **get_functionDescriptor**( )
es	async **get_functionDescriptor**( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

Retourne :
un identifiant de type YFUN_DESCR.

Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

cellular→get_functionId()
cellular→functionId()

YCellular

Retourne l'identifiant matériel de l'interface cellulaire, sans référence au module.

```
js      function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
dnp     string get_functionId( )
java    String get_functionId( )
py      get_functionId( )
php     function get_functionId( )
es      async get_functionId( )
```

Par example `relay1`.

Retourne :
une chaîne de caractères identifiant l'interface cellulaire (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

cellular→get_hardwareId() YCellular
cellular→hardwareId()

Retourne l'identifiant matériel unique de l'interface cellulaire au format SERIAL.FUNCTIONID.

js	function **get_hardwareId**( )
cpp	string **get_hardwareId**( )
m	-(NSString*) hardwareId
vb	function **get_hardwareId**( ) As String
cs	string **get_hardwareId**( )
dnp	string **get_hardwareId**( )
java	String **get_hardwareId**( )
py	**get_hardwareId**( )
php	function **get_hardwareId**( )
es	async **get_hardwareId**( )

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'interface cellulaire (par example RELAYLO1-123456.relay1).

Retourne :
une chaîne de caractères identifiant l'interface cellulaire (ex: RELAYLO1-123456.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

cellular→get_imsi()
cellular→imsi()

YCellular

Retourne une string opaque si un code PIN a été configuré dans le module pour accéder à la carte SIM, ou une chaîne vide il n'a pas été configuré ou si la SIM a rejeté le code indiqué.

```
js      function get_imsi( )
cpp     string get_imsi( )
m       -(NSString*) imsi
pas     string get_imsi( ): string
vb      function get_imsi( ) As String
cs      string get_imsi( )
dnp     string get_imsi( )
java    String get_imsi( )
uwp     async Task<string> get_imsi( )
py      get_imsi( )
php     function get_imsi( )
es      async get_imsi( )
cmd     YCellular target get_imsi
```

Retourne :

une chaîne de caractères représentant une string opaque si un code PIN a été configuré dans le module pour accéder à la carte SIM, ou une chaîne vide il n'a pas été configuré ou si la SIM a rejeté le code indiqué

En cas d'erreur, déclenche une exception ou retourne `Y_IMSI_INVALID`.

cellular→get_linkQuality()
cellular→linkQuality()

YCellular

Retourne la qualité de la connection, exprimée en pourcents.

```
js    function get_linkQuality( )
cpp   int get_linkQuality( )
m     -(int) linkQuality
pas   LongInt get_linkQuality( ): LongInt
vb    function get_linkQuality( ) As Integer
cs    int get_linkQuality( )
dnp   int get_linkQuality( )
java  int get_linkQuality( )
uwp   async Task<int> get_linkQuality( )
py    get_linkQuality( )
php   function get_linkQuality( )
es    async get_linkQuality( )
cmd   YCellular target get_linkQuality
```

Retourne :
un entier représentant la qualité de la connection, exprimée en pourcents

En cas d'erreur, déclenche une exception ou retourne `Y_LINKQUALITY_INVALID`.

cellular→get_lockedOperator() YCellular
cellular→lockedOperator()

Retourne le nom de l'opérateur de réseau cellulaire à utiliser exclusivement, si le choix automatique est désactivé, ou une chaîne vide si la carte SIM sélectionne automatiquement l'opérateur selon ceux disponibles.

```
js      function get_lockedOperator( )
cpp     string get_lockedOperator( )
m       -(NSString*) lockedOperator
pas     string get_lockedOperator( ): string
vb      function get_lockedOperator( ) As String
cs      string get_lockedOperator( )
dnp     string get_lockedOperator( )
java    String get_lockedOperator( )
uwp     async Task<string> get_lockedOperator( )
py      get_lockedOperator( )
php     function get_lockedOperator( )
es      async get_lockedOperator( )
cmd     YCellular target get_lockedOperator
```

Retourne :

une chaîne de caractères représentant le nom de l'opérateur de réseau cellulaire à utiliser exclusivement, si le choix automatique est désactivé, ou une chaîne vide si la carte SIM sélectionne automatiquement l'opérateur selon ceux disponibles

En cas d'erreur, déclenche une exception ou retourne `Y_LOCKEDOPERATOR_INVALID`.

cellular→get_logicalName() YCellular
cellular→logicalName()

Retourne le nom logique de l'interface cellulaire.

```
js      function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     string get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
dnp     string get_logicalName( )
java    String get_logicalName( )
uwp     async Task<string> get_logicalName( )
py      get_logicalName( )
php     function get_logicalName( )
es      async get_logicalName( )
cmd     YCellular target get_logicalName
```

Retourne :
une chaîne de caractères représentant le nom logique de l'interface cellulaire.

En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

cellular→get_message()
cellular→message()

YCellular

Retourne le dernier message de diagnostique de l'interface au réseau sans fil.

```
js      function get_message( )
cpp     string get_message( )
m       -(NSString*) message
pas     string get_message( ): string
vb      function get_message( ) As String
cs      string get_message( )
dnp     string get_message( )
java    String get_message( )
uwp     async Task<string> get_message( )
py      get_message( )
php     function get_message( )
es      async get_message( )
cmd     YCellular target get_message
```

Retourne :
une chaîne de caractères représentant le dernier message de diagnostique de l'interface au réseau sans fil

En cas d'erreur, déclenche une exception ou retourne `Y_MESSAGE_INVALID`.

cellular→get_module()
cellular→module()

YCellular

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function **get_module**( )
cpp	YModule * **get_module**( )
m	-(YModule*) module
pas	TYModule **get_module**( ): TYModule
vb	function **get_module**( ) As YModule
cs	YModule **get_module**( )
dnp	YModuleProxy **get_module**( )
java	YModule **get_module**( )
py	**get_module**( )
php	function **get_module**( )
es	async **get_module**( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Retourne :
une instance de YModule

cellular→get_module_async()
cellular→module_async()

YCellular

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

cellular→get_pin()
cellular→pin()

YCellular

Retourne une string opaque si un code PIN a été configuré dans le module pour accéder à la carte SIM, ou une chaîne vide il n'a pas été configuré ou si la SIM a rejeté le code indiqué.

```
js      function get_pin( )
cpp     string get_pin( )
m       -(NSString*) pin
pas     string get_pin( ): string
vb      function get_pin( ) As String
cs      string get_pin( )
dnp     string get_pin( )
java    String get_pin( )
uwp     async Task<string> get_pin( )
py      get_pin( )
php     function get_pin( )
es      async get_pin( )
cmd     YCellular target get_pin
```

Retourne :

une chaîne de caractères représentant une string opaque si un code PIN a été configuré dans le module pour accéder à la carte SIM, ou une chaîne vide il n'a pas été configuré ou si la SIM a rejeté le code indiqué

En cas d'erreur, déclenche une exception ou retourne `Y_PIN_INVALID`.

cellular→get_pingInterval()
cellular→pingInterval()

YCellular

Retourne l'intervalle entre les tests de connectivité spontanés, en secondes.

```
js    function get_pingInterval( )
cpp   int get_pingInterval( )
m     -(int) pingInterval
pas   LongInt get_pingInterval( ): LongInt
vb    function get_pingInterval( ) As Integer
cs    int get_pingInterval( )
dnp   int get_pingInterval( )
java  int get_pingInterval( )
uwp   async Task<int> get_pingInterval( )
py    get_pingInterval( )
php   function get_pingInterval( )
es    async get_pingInterval( )
cmd   YCellular target get_pingInterval
```

Retourne :
un entier représentant l'intervalle entre les tests de connectivité spontanés, en secondes

En cas d'erreur, déclenche une exception ou retourne `Y_PINGINTERVAL_INVALID`.

cellular→get_serialNumber() YCellular
cellular→serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

```
js      function get_serialNumber( )
cpp     string get_serialNumber( )
m       -(NSString*) serialNumber
pas     string get_serialNumber( ): string
vb      function get_serialNumber( ) As String
cs      string get_serialNumber( )
dnp     string get_serialNumber( )
java    String get_serialNumber( )
uwp     async Task<string> get_serialNumber( )
py      get_serialNumber( )
php     function get_serialNumber( )
es      async get_serialNumber( )
cmd     YCellular target get_serialNumber
```

Retourne :

: une chaîne de caractères représentant le numéro de série du module, préprogrammé en usine.

En cas d'erreur, déclenche une exception ou retourne YModule.SERIALNUMBER_INVALID.

cellular→get_userData()
cellular→userData()

YCellular

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

```
js      function get_userData( )
cpp     void * get_userData( )
m       -(id) userData
pas     Tobject get_userData( ): Tobject
vb      function get_userData( ) As Object
cs      object get_userData( )
java    Object get_userData( )
py      get_userData( )
php     function get_userData( )
es      async get_userData( )
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Retourne :
l'objet stocké précédemment par l'appelant.

cellular→isOnline() YCellular

Vérifie si le module hébergeant l'interface cellulaire est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     boolean isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
dnp     bool isOnline( )
java    boolean isOnline( )
py      isOnline( )
php     function isOnline( )
es      async isOnline( )
```

Si les valeurs des attributs en cache de l'interface cellulaire sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Retourne :
`true` si l'interface cellulaire est joignable, `false` sinon

cellular→isOnline_async() YCellular

Vérifie si le module hébergeant l'interface cellulaire est joignable, sans déclencher d'erreur.

```
js  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'interface cellulaire sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

cellular→isReadOnly() YCellular

Test si la fonction est en lecture seule.

cpp	bool **isReadOnly**( )
m	-(bool) **isReadOnly**
pas	boolean **isReadOnly**( ): boolean
vb	function **isReadOnly**( ) As Boolean
cs	bool **isReadOnly**( )
dnp	bool **isReadOnly**( )
java	boolean **isReadOnly**( )
uwp	async Task<bool> **isReadOnly**( )
py	**isReadOnly**( )
php	function **isReadOnly**( )
es	async **isReadOnly**( )
cmd	YCellular **target isReadOnly**

Retourne vrais si la fonction est protégé en ecriture ou que la fontion n'est pas disponible.

Retourne :
`true` si la fonction est protégé en ecriture ou que la fontion n'est pas disponible

cellular→load() YCellular

Met en cache les valeurs courantes de l'interface cellulaire, avec une durée de validité spécifiée.

```
js      function load( msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (u64) msValidity
pas     YRETCODE load( msValidity: u64): YRETCODE
vb      function load( ByVal msValidity As Long) As YRETCODE
cs      YRETCODE load( ulong msValidity)
java    int load( long msValidity)
py      load( msValidity)
php     function load( $msValidity)
es      async load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→loadAttribute() YCellular

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

```
js      function loadAttribute( attrName)
cpp     string loadAttribute( string attrName)
m       -(NSString*) loadAttribute : (NSString*) attrName
pas     string loadAttribute( attrName: string): string
vb      function loadAttribute( ) As String
cs      string loadAttribute( string attrName)
dnp     string loadAttribute( string attrName)
java    String loadAttribute( String attrName)
uwp     async Task<string> loadAttribute( string attrName)
py      loadAttribute( attrName)
php     function loadAttribute( $attrName)
es      async loadAttribute( attrName)
```

Paramètres :

attrName le nom de l'attribut désiré

Retourne :

une chaîne de caractères représentant la valeur actuelle de l'attribut.

En cas d'erreur, déclenche une exception ou retourne un chaîne vide.

cellular→load_async() YCellular

Met en cache les valeurs courantes de l'interface cellulaire, avec une durée de validité spécifiée.

```
js  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

msValidity	un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
callback	fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)
context	contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

cellular→muteValueCallbacks() YCellular

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

js	function **muteValueCallbacks**( )
cpp	int **muteValueCallbacks**( )
m	-(int) **muteValueCallbacks**
pas	LongInt **muteValueCallbacks**( ): LongInt
vb	function **muteValueCallbacks**( ) As Integer
cs	int **muteValueCallbacks**( )
dnp	int **muteValueCallbacks**( )
java	int **muteValueCallbacks**( )
uwp	async Task<int> **muteValueCallbacks**( )
py	**muteValueCallbacks**( )
php	function **muteValueCallbacks**( )
es	async **muteValueCallbacks**( )
cmd	YCellular **target muteValueCallbacks**

Vous pouvez utiliser cette fonction pour économiser la bande passante et le CPU sur les machines de faible puissance, ou pour éviter le déclanchement de callbacks HTTP. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→nextCellular() YCellular

Continue l'énumération des interfaces réseau cellulaire commencée à l'aide de `yFirstCellular()` Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les interfaces réseau cellulaire sont retournés.

```
js      function nextCellular( )
cpp     YCellular * nextCellular( )
m       -(YCellular*) nextCellular
pas     TYCellular nextCellular( ): TYCellular
vb      function nextCellular( ) As YCellular
cs      YCellular nextCellular( )
java    YCellular nextCellular( )
uwp     YCellular nextCellular( )
py      nextCellular( )
php     function nextCellular( )
es      nextCellular( )
```

Si vous souhaitez retrouver une interface cellulaire spécifique, utilisez `Cellular.findCellular()` avec un hardwareID ou un nom logique.

Retourne :
un pointeur sur un objet `YCellular` accessible en ligne, ou `null` lorsque l'énumération est terminée.

cellular→quickCellSurvey() YCellular

Retourne la liste d'identifiants pour les antennes GSM à proximité, telle que requise pour géolocaliser rapidement le module.

```
js      function quickCellSurvey( )
cpp     vector<YCellRecord> quickCellSurvey( )
m       -(NSMutableArray*) quickCellSurvey
pas     TYCellRecordArray quickCellSurvey( ): TYCellRecordArray
vb      function quickCellSurvey( ) As List
cs      List<YCellRecord> quickCellSurvey( )
dnp     YCellRecordProxy[] quickCellSurvey( )
java    ArrayList<YCellRecord> quickCellSurvey( )
uwp     async Task<List<YCellRecord>> quickCellSurvey( )
py      quickCellSurvey( )
php     function quickCellSurvey( )
es      async quickCellSurvey( )
cmd     YCellular target quickCellSurvey
```

La première antenne listée est la cellule active, et les suivantes sont les cellules voisines listée par la cellule active.

Retourne :
une liste de `YCellRecord`.

cellular→registerValueCallback() YCellular

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

```
js    function registerValueCallback( callback)
cpp   int registerValueCallback( YCellularValueCallback callback)
m     -(int) registerValueCallback : (YCellularValueCallback) callback
pas   LongInt registerValueCallback( callback: TYCellularValueCallback): LongInt
vb    function registerValueCallback( ) As Integer
cs    int registerValueCallback( ValueCallback callback)
java  int registerValueCallback( UpdateCallback callback)
uwp   async Task<int> registerValueCallback( ValueCallback callback)
py    registerValueCallback( callback)
php   function registerValueCallback( $callback)
es    async registerValueCallback( callback)
```

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

Paramètres :

callback la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

cellular→sendPUK() YCellular

Envoie le code PUK à la carte SIM pour la débloquer après trois échecs consécutifs de code PIN, et établit un nouveau code PIN dans la SIM.

```
js    function sendPUK( puk, newPin)
cpp   int sendPUK( string puk, string newPin)
m     -(int) sendPUK : (NSString*) puk : (NSString*) newPin
pas   LongInt sendPUK( puk: string, newPin: string): LongInt
vb    function sendPUK( ) As Integer
cs    int sendPUK( string puk, string newPin)
dnp   int sendPUK( string puk, string newPin)
java  int sendPUK( String puk, String newPin)
uwp   async Task<int> sendPUK( string puk, string newPin)
py    sendPUK( puk, newPin)
php   function sendPUK( $puk, $newPin)
es    async sendPUK( puk, newPin)
cmd   YCellular target sendPUK puk newPin
```

Seules dix tentatives consécutives de déblocage sont possibles: après dix tentatives infructueuses, la carte SIM sera définitivement inutilisable. Après avoir appelé cette fonction, vous devrez aussi appeler la méthode `set_pin()` pour indiquer au YoctoHub le nouveau PIN à utiliser dans le futur.

Paramètres :

puk code PUK de la carte SIM

newPin nouveau code PIN à configurer dans la carte SIM

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→set_airplaneMode()
cellular→setAirplaneMode()

YCellular

Modifie l'état du mode avion (radio désactivée).

```
js    function set_airplaneMode( newval)
cpp   int set_airplaneMode( Y_AIRPLANEMODE_enum newval)
m     -(int) setAirplaneMode : (Y_AIRPLANEMODE_enum) newval
pas   integer set_airplaneMode( newval: Integer): integer
vb    function set_airplaneMode( ByVal newval As Integer) As Integer
cs    int set_airplaneMode( int newval)
dnp   int set_airplaneMode( int newval)
java  int set_airplaneMode( int newval)
uwp   async Task<int> set_airplaneMode( int newval)
py    set_airplaneMode( newval)
php   function set_airplaneMode( $newval)
es    async set_airplaneMode( newval)
cmd   YCellular target set_airplaneMode newval
```

Paramètres :

newval soit `Y_AIRPLANEMODE_OFF`, soit `Y_AIRPLANEMODE_ON`, selon l'état du mode avion (radio désactivée)

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→set_apn() YCellular
cellular→setApn()

Retourne le nom du point d'accès (APN) à utiliser, si nécessaire.

```
js    function set_apn( newval)
cpp   int set_apn( string newval)
m     -(int) setApn : (NSString*) newval
pas   integer set_apn( newval: string): integer
vb    function set_apn( ByVal newval As String) As Integer
cs    int set_apn( string newval)
dnp   int set_apn( string newval)
java  int set_apn( String newval)
uwp   async Task<int> set_apn( string newval)
py    set_apn( newval)
php   function set_apn( $newval)
es    async set_apn( newval)
cmd   YCellular target set_apn newval
```

Lorsque l'APN est vide, celui proposé par l'opérateur cellulaire est utilisée. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→set_apnAuth()
cellular→setApnAuth()

YCellular

Configure les paramètres d'identification pour se connecter à l'APN.

```
js    function set_apnAuth( username, password)
cpp   int set_apnAuth( string username, string password)
m     -(int) setApnAuth : (NSString*) username
                        : (NSString*) password
pas   LongInt set_apnAuth( username: string, password: string): LongInt
vb    function set_apnAuth( ) As Integer
cs    int set_apnAuth( string username, string password)
dnp   int set_apnAuth( string username, string password)
java  int set_apnAuth( String username, String password)
uwp   async Task<int> set_apnAuth( string username, string password)
py    set_apnAuth( username, password)
php   function set_apnAuth( $username, $password)
es    async set_apnAuth( username, password)
cmd   YCellular target set_apnAuth username password
```

Les protocoles PAP et CHAP sont tous deux supportés.

Paramètres :

username nom d'utilisateur

password mot de passe

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→set_dataReceived() YCellular
cellular→setDataReceived()

Modifie la valeur du compteur d'octets reçus.

```
js    function set_dataReceived( newval)
cpp   int set_dataReceived( int newval)
m     -(int) setDataReceived : (int) newval
pas   integer set_dataReceived( newval: LongInt): integer
vb    function set_dataReceived( ByVal newval As Integer) As Integer
cs    int set_dataReceived( int newval)
dnp   int set_dataReceived( int newval)
java  int set_dataReceived( int newval)
uwp   async Task<int> set_dataReceived( int newval)
py    set_dataReceived( newval)
php   function set_dataReceived( $newval)
es    async set_dataReceived( newval)
cmd   YCellular target set_dataReceived newval
```

Paramètres :

newval un entier représentant la valeur du compteur d'octets reçus

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→set_dataSent()
cellular→setDataSent()

YCellular

Modifie la valeur du compteur d'octets envoyés.

```
js    function set_dataSent( newval)
cpp   int set_dataSent( int newval)
m     -(int) setDataSent : (int) newval
pas   integer set_dataSent( newval: LongInt): integer
vb    function set_dataSent( ByVal newval As Integer) As Integer
cs    int set_dataSent( int newval)
dnp   int set_dataSent( int newval)
java  int set_dataSent( int newval)
uwp   async Task<int> set_dataSent( int newval)
py    set_dataSent( newval)
php   function set_dataSent( $newval)
es    async set_dataSent( newval)
cmd   YCellular target set_dataSent newval
```

Paramètres :

newval un entier représentant la valeur du compteur d'octets envoyés

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→set_enableData() YCellular
cellular→setEnableData()

Modifie la condition dans laquelle le service de données IP (GRPS) doit être activé.

```
js      function set_enableData( newval)
cpp     int set_enableData( Y_ENABLEDATA_enum newval)
m       -(int) setEnableData : (Y_ENABLEDATA_enum) newval
pas     integer set_enableData( newval: Integer): integer
vb      function set_enableData( ByVal newval As Integer) As Integer
cs      int set_enableData( int newval)
dnp     int set_enableData( int newval)
java    int set_enableData( int newval)
uwp     async Task<int> set_enableData( int newval)
py      set_enableData( newval)
php     function set_enableData( $newval)
es      async set_enableData( newval)
cmd     YCellular target set_enableData newval
```

Le service peut être soit complètement désactivé, soit limité au réseau de de l'émetteur de la carte SIM, soit être activé pour tous les réseaux en partenariat avec la carte SIM (roaming). Attention, l'utilisation de données en roaming peut conduire à des coûts de télécommunication exhorbitants !

Lorsque le service de donnée n'est pas actif, seules les communications par SMS sont possibles. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une valeur parmi `Y_ENABLEDATA_HOMENETWORK`, `Y_ENABLEDATA_ROAMING`, `Y_ENABLEDATA_NEVER` et `Y_ENABLEDATA_NEUTRALITY` représentant la condition dans laquelle le service de données IP (GRPS) doit être activé

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→set_lockedOperator() YCellular
cellular→setLockedOperator()

Modifie le nom de l'opérateur de réseau cellulaire à utiliser.

```
js    function set_lockedOperator( newval)
cpp   int set_lockedOperator( string newval)
m     -(int) setLockedOperator : (NSString*) newval
pas   integer set_lockedOperator( newval: string): integer
vb    function set_lockedOperator( ByVal newval As String) As Integer
cs    int set_lockedOperator( string newval)
dnp   int set_lockedOperator( string newval)
java  int set_lockedOperator( String newval)
uwp   async Task<int> set_lockedOperator( string newval)
py    set_lockedOperator( newval)
php   function set_lockedOperator( $newval)
es    async set_lockedOperator( newval)
cmd   YCellular target set_lockedOperator newval
```

Si le nom est une chaîne vide, le choix sera fait automatiquement selon la carte SIM. Sinon, seul l'opérateur choisi sera utilisé. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom de l'opérateur de réseau cellulaire à utiliser

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→set_logicalName() YCellular
cellular→setLogicalName()

Modifie le nom logique de l'interface cellulaire.

js	function **set_logicalName**(**newval**)
cpp	int **set_logicalName**(string **newval**)
m	-(int) setLogicalName : (NSString*) **newval**
pas	integer **set_logicalName**(**newval**: string): integer
vb	function **set_logicalName**(ByVal **newval** As String) As Integer
cs	int **set_logicalName**(string **newval**)
dnp	int **set_logicalName**(string **newval**)
java	int **set_logicalName**(String **newval**)
uwp	async Task<int> **set_logicalName**(string **newval**)
py	**set_logicalName**(**newval**)
php	function **set_logicalName**(**$newval**)
es	async **set_logicalName**(**newval**)
cmd	YCellular **target set_logicalName newval**

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom logique de l'interface cellulaire.

Retourne :

`YAPI_SUCCESS` si l'appel se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→set_pin()
cellular→setPin()

YCellular

Modifie le code PIN utilisé par le module pour accéder à la carte SIM.

```
js      function set_pin( newval)
cpp     int set_pin( string newval)
m       -(int) setPin : (NSString*) newval
pas     integer set_pin( newval: string): integer
vb      function set_pin( ByVal newval As String) As Integer
cs      int set_pin( string newval)
dnp     int set_pin( string newval)
java    int set_pin( String newval)
uwp     async Task<int> set_pin( string newval)
py      set_pin( newval)
php     function set_pin( $newval)
es      async set_pin( newval)
cmd     YCellular target set_pin newval
```

Cette fonction ne change pas le code sur la carte SIM elle-même, mais uniquement le paramètre utilisé par le module pour essayer d'en obtenir l'accès. Si le code SIM ne fonctionne pas dès le premier essai, il sera automatiquement oublié et un message "Enter SIM PIN" apparaîtra dans l'attribut 'message'. Il faudra alors appeler à nouveau cette méthode avec le bon code PIN. Après trois essais infructueux consécutifs le message devient "Enter SIM PUK" et il faut alors entrer le code PUK de la carte SIM avec la méthode `sendPUK`.

N'oubliez pas d'appeler la méthode `saveToFlash()` du module pour que le paramètre soit sauvegardé dans la flash.

Paramètres :

newval une chaîne de caractères représentant le code PIN utilisé par le module pour accéder à la carte SIM

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→set_pingInterval()
cellular→setPingInterval()

YCellular

Modifie l'intervalle entre les tests de connectivité spontanés, en secondes.

js	function **set_pingInterval**(**newval**)
cpp	int **set_pingInterval**(int **newval**)
m	-(int) setPingInterval : (int) **newval**
pas	integer **set_pingInterval**(**newval**: LongInt): integer
vb	function **set_pingInterval**(ByVal **newval** As Integer) As Integer
cs	int **set_pingInterval**(int **newval**)
dnp	int **set_pingInterval**(int **newval**)
java	int **set_pingInterval**(int **newval**)
uwp	async Task<int> **set_pingInterval**(int **newval**)
py	**set_pingInterval**(**newval**)
php	function **set_pingInterval**(**$newval**)
es	async **set_pingInterval**(**newval**)
cmd	YCellular **target set_pingInterval newval**

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant l'intervalle entre les tests de connectivité spontanés, en secondes

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→set_userData()
cellular→setUserData()

YCellular

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js      function set_userData( data)
cpp     void set_userData( void * data)
m       -(void) setUserData : (id) data
pas     set_userData( data: Tobject)
vb      procedure set_userData( ByVal data As Object)
cs      void set_userData( object data)
java    void set_userData( Object data)
py      set_userData( data)
php     function set_userData( $data)
es      async set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Paramètres :

data objet quelconque à mémoriser

cellular→unmuteValueCallbacks() YCellular

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

js	function **unmuteValueCallbacks**( )
cpp	int **unmuteValueCallbacks**( )
m	-(int) **unmuteValueCallbacks**
pas	LongInt **unmuteValueCallbacks**( ): LongInt
vb	function **unmuteValueCallbacks**( ) As Integer
cs	int **unmuteValueCallbacks**( )
dnp	int **unmuteValueCallbacks**( )
java	int **unmuteValueCallbacks**( )
uwp	async Task<int> **unmuteValueCallbacks**( )
py	**unmuteValueCallbacks**( )
php	function **unmuteValueCallbacks**( )
es	async **unmuteValueCallbacks**( )
cmd	YCellular **target unmuteValueCallbacks**

Cette fonction annule un précédent appel à `muteValueCallbacks()`. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

cellular→wait_async() YCellular

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js  function wait_async( callback, context)
es  wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

Paramètres :

callback fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout.

8.3. La classe YNetwork

Interface pour intéragir avec les interfaces réseau, disponibles par exemple dans le YoctoHub-Ethernet, le YoctoHub-GSM-3G-EU, le YoctoHub-GSM-3G-NA et le YoctoHub-Wireless-g

La classe `YNetwork` permet de contrôler les paramètres TCP/IP des modules Yoctopuce dotés d'une interface réseau.

Pour utiliser les fonctions décrites ici, vous devez inclure:

es	in HTML: <script src="../../lib/yocto_network.js"></script> in node.js: require('yoctolib-es2017/yocto_network.js');
js	<script type='text/javascript' src='yocto_network.js'></script>
cpp	#include "yocto_network.h"
m	#import "yocto_network.h"
pas	uses yocto_network;
vb	yocto_network.vb
cs	yocto_network.cs
dnp	import YoctoProxyAPI.YNetworkProxy
java	import com.yoctopuce.YoctoAPI.YNetwork;
uwp	import com.yoctopuce.YoctoAPI.YNetwork;
py	from yocto_network import *
php	require_once('yocto_network.php');
vi	YNetwork.vi

Fonction globales

YNetwork.FindNetwork(**func**)

Permet de retrouver une interface réseau d'après un identifiant donné.

YNetwork.FindNetworkInContext(**yctx**, **func**)

Permet de retrouver une interface réseau d'après un identifiant donné dans un Context YAPI.

YNetwork.FirstNetwork()

Commence l'énumération des interfaces réseau accessibles par la librairie.

YNetwork.FirstNetworkInContext(**yctx**)

Commence l'énumération des interfaces réseau accessibles par la librairie.

YNetwork.GetSimilarFunctions()

Enumère toutes les fonctions de type Network disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).

Propriétés des objets `YNetworkProxy`

network→AdminPassword *[modifiable]*

Chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide.

network→AdvertisedValue *[lecture seule]*

Courte chaîne de caractères représentant l'état courant de la fonction.

network→CallbackCredentials *[modifiable]*

Version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide.

network→CallbackEncoding *[modifiable]*

Encodage à utiliser pour représenter les valeurs notifiées par callback.

network→CallbackInitialDelay *[modifiable]*

Attente initiale avant la première notification par callback, en secondes.

network→CallbackMaxDelay *[modifiable]*

Attente entre deux callback HTTP lorsque rien n'est à signaler, en secondes.

network→CallbackMethod *[modifiable]*

Méthode HTTP à utiliser pour signaler les changements d'état par callback.

network→CallbackMinDelay *[modifiable]*

Attente minimale entre deux callbacks HTTP, en secondes.

network→CallbackSchedule *[modifiable]*

Planification des callbacks HTTP, sous forme de chaîne de caractères.

network→CallbackUrl *[modifiable]*

Adresse (URL) de callback à notifier lors de changement d'état significatifs.

network→DefaultPage *[modifiable]*

Page HTML à envoyer pour l'URL "/" **Modifiable**.

network→Discoverable *[modifiable]*

état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).

network→FriendlyName *[lecture seule]*

Identifiant global de la fonction au format `NOM_MODULE.NOM_FONCTION`.

network→FunctionId *[lecture seule]*

Identifiant matériel de l'interface réseau, sans référence au module.

network→HardwareId *[lecture seule]*

Identifiant matériel unique de la fonction au format `SERIAL.FUNCTIONID`.

network→HttpPort *[modifiable]*

Port TCP utilisé pour l'interface Web du hub.

network→IpAddress *[lecture seule]*

Adresse IP utilisée par le module Yoctopuce.

network→IsOnline *[lecture seule]*

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

network→LogicalName *[modifiable]*

Nom logique de la fonction.

network→MacAddress *[lecture seule]*

Adresse MAC de l'interface réseau, unique pour chaque module.

network→NtpServer *[modifiable]*

Adresse IP du serveur de NTP à utiliser pour maintenir le module à l'heure.

network→PrimaryDNS *[modifiable]*

Adresse IP du serveur de noms primaire que le module doit utiliser.

network→Readiness *[lecture seule]*

état de fonctionnement atteint par l'interface réseau.

network→SecondaryDNS *[modifiable]*

Adresse IP du serveur de noms secondaire que le module doit utiliser.

network→SerialNumber *[lecture seule]*

Numéro de série du module, préprogrammé en usine.

network→UserPassword *[modifiable]*

Chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide.

network→WwwWatchdogDelay *[modifiable]*

Durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

Méthodes des objets `YNetwork`

network→callbackLogin(username, password)

Contacte le callback de notification et sauvegarde un laisser-passer pour s'y connecter.

network→clearCache()

Invalide le cache.

network→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'interface réseau au format TYPE(NAME)=SERIAL.FUNCTIONID.

network→get_adminPassword()

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide.

network→get_advertisedValue()

Retourne la valeur courante de l'interface réseau (pas plus de 6 caractères).

network→get_callbackCredentials()

Retourne une version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide.

network→get_callbackEncoding()

Retourne l'encodage à utiliser pour représenter les valeurs notifiées par callback.

network→get_callbackInitialDelay()

Retourne l'attente initiale avant la première notification par callback, en secondes.

network→get_callbackMaxDelay()

Retourne l'attente entre deux callback HTTP lorsque rien n'est à signaler, en secondes.

network→get_callbackMethod()

Retourne la méthode HTTP à utiliser pour signaler les changements d'état par callback.

network→get_callbackMinDelay()

Retourne l'attente minimale entre deux callbacks HTTP, en secondes.

network→get_callbackSchedule()

Retourne la planification des callbacks HTTP, sous forme de chaîne de caractères.

network→get_callbackUrl()

Retourne l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

network→get_defaultPage()

Retourne la page HTML à envoyer pour l'URL "/"

network→get_discoverable()

Retourne l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).

network→get_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

network→get_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

network→get_friendlyName()

Retourne un identifiant global de l'interface réseau au format NOM_MODULE.NOM_FONCTION.

network→get_functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

network→get_functionId()

Retourne l'identifiant matériel de l'interface réseau, sans référence au module.

network→get_hardwareId()

Retourne l'identifiant matériel unique de l'interface réseau au format SERIAL.FUNCTIONID.

network→get_httpPort()

Retourne le port TCP utilisé pour l'interface Web du hub.

network→get_ipAddress()

Retourne l'adresse IP utilisée par le module Yoctopuce.

network→get_ipConfig()

Retourne la configuration IP de l'interface réseau.

network→get_logicalName()

Retourne le nom logique de l'interface réseau.

network→get_macAddress()

Retourne l'adresse MAC de l'interface réseau, unique pour chaque module.

network→get_module()

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

network→get_module_async(**callback**, **context**)

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

network→get_ntpServer()

Retourne l'adresse IP du serveur de NTP à utiliser pour maintenir le module à l'heure.

network→get_poeCurrent()

Retourne le courant consommé par le module depuis Power-over-Ethernet (PoE), en milliampères.

network→get_primaryDNS()

Retourne l'adresse IP du serveur de noms primaire que le module doit utiliser.

network→get_readiness()

Retourne l'état de fonctionnement atteint par l'interface réseau.

network→get_router()

Retourne l'adresse IP du routeur (passerelle) utilisé par le module (*default gateway*).

network→get_secondaryDNS()

Retourne l'adresse IP du serveur de noms secondaire que le module doit utiliser.

network→get_serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

network→get_subnetMask()

Retourne le masque de sous-réseau utilisé par le module.

network→get_userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

network→get_userPassword()

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide.

network→get_wwwWatchdogDelay()

Retourne la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

network→isOnline()

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

network→isOnline_async(**callback**, **context**)

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

network→isReadOnly()

Test si la fonction est en lecture seule.

network→load(**msValidity**)

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

network→loadAttribute(attrName)

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

network→load_async(msValidity, callback, context)

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

network→muteValueCallbacks()

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

network→nextNetwork()

Continue l'énumération des interfaces réseau commencée à l'aide de `yFirstNetwork()` Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les interfaces réseau sont retournés.

network→ping(host)

Ping l'adresse choisie pour vérifier la connexion réseau.

network→registerValueCallback(callback)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

network→set_adminPassword(newval)

Modifie le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module.

network→set_callbackCredentials(newval)

Modifie le laisser-passer pour se connecter à l'adresse de callback.

network→set_callbackEncoding(newval)

Modifie l'encodage à utiliser pour représenter les valeurs notifiées par callback.

network→set_callbackInitialDelay(newval)

Modifie l'attente initiale avant la première notification par callback, en secondes.

network→set_callbackMaxDelay(newval)

Modifie l'attente entre deux callback HTTP lorsque rien n'est à signaler.

network→set_callbackMethod(newval)

Modifie la méthode HTTP à utiliser pour signaler les changements d'état par callback.

network→set_callbackMinDelay(newval)

Modifie l'attente minimale entre deux callbacks HTTP, en secondes.

network→set_callbackSchedule(newval)

Modifie la planification des callbacks HTTP, sous forme de chaîne de caractères.

network→set_callbackUrl(newval)

Modifie l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

network→set_defaultPage(newval)

Modifie la page HTML par défaut du hub.

network→set_discoverable(newval)

Modifie l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).

network→set_httpPort(newval)

Modifie le port TCP utilisé pour l'interface Web du hub.

network→set_logicalName(newval)

Modifie le nom logique de l'interface réseau.

network→set_ntpServer(newval)

Modifie l'adresse IP du serveur NTP que le module doit utiliser.

network→set_periodicCallbackSchedule(interval, offset)

Configure la planification de callbacks HTTP périodiques (fonction simplifiée).

network→set_primaryDNS(newval)

Modifie l'adresse IP du serveur de noms primaire que le module doit utiliser.

network→set_secondaryDNS(newval)

Modifie l'adresse IP du serveur de nom secondaire que le module doit utiliser.

network→set_userData(data)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

network→set_userPassword(newval)

Modifie le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module.

network→set_wwwWatchdogDelay(newval)

Modifie la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

network→triggerCallback()

Déclenche un callback HTTP rapidement.

network→unmuteValueCallbacks()

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

network→useDHCP(fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)

Modifie la configuration de l'interface réseau pour utiliser une adresse assignée automatiquement par le serveur DHCP.

network→useDHCPauto()

Modifie la configuration de l'interface réseau pour utiliser une adresse assignée automatiquement par le serveur DHCP.

network→useStaticIP(ipAddress, subnetMaskLen, router)

Modifie la configuration de l'interface réseau pour utiliser une adresse IP assignée manuellement (adresse IP statique).

network→wait_async(callback, context)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

YNetwork.FindNetwork() YNetwork
YNetwork.FindNetwork()

Permet de retrouver une interface réseau d'après un identifiant donné.

```
js      function yFindNetwork( func)
cpp     YNetwork* yFindNetwork( string func)
m       +(YNetwork*) FindNetwork : (NSString*) func
pas     TYNetwork yFindNetwork( func: string): TYNetwork
vb      function yFindNetwork( ByVal func As String) As YNetwork
cs      static YNetwork FindNetwork( string func)
dnp     static YNetworkProxy FindNetwork( string func)
java    static YNetwork FindNetwork( String func)
uwp     static YNetwork FindNetwork( string func)
py      FindNetwork( func)
php     function yFindNetwork( $func)
es      static FindNetwork( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'interface réseau soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YNetwork.isOnline()` pour tester si l'interface réseau est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Si un appel à la méthode is_online() de cet objet renvoie FAUX alors que vous êtes sûr que le module correspondant est bien branché, vérifiez que vous n'avez pas oublié d'appeler registerHub() à l'initialisation de de l'application.

Paramètres :

func une chaîne de caractères qui référence l'interface réseau sans ambiguïté, par exemple `YHUBETH1.network`.

Retourne :

un objet de classe `YNetwork` qui permet ensuite de contrôler l'interface réseau.

YNetwork.FindNetworkInContext() YNetwork
YNetwork.FindNetworkInContext()

Permet de retrouver une interface réseau d'après un identifiant donné dans un Context YAPI.

java static YNetwork **FindNetworkInContext**(YAPIContext **yctx**, String **func**)
uwp static YNetwork **FindNetworkInContext**(YAPIContext **yctx**, string **func**)
es static **FindNetworkInContext**(**yctx**, **func**)

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'interface réseau soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YNetwork.isOnline()` pour tester si l'interface réseau est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Paramètres :

yctx un contexte YAPI

func une chaîne de caractères qui référence l'interface réseau sans ambiguïté, par exemple `YHUBETH1.network`.

Retourne :

un objet de classe `YNetwork` qui permet ensuite de contrôler l'interface réseau.

YNetwork.FirstNetwork() YNetwork
YNetwork.FirstNetwork()

Commence l'énumération des interfaces réseau accessibles par la librairie.

js	function **yFirstNetwork**( )
cpp	YNetwork * **yFirstNetwork**( )
m	+(YNetwork*) **FirstNetwork**
pas	TYNetwork **yFirstNetwork**( ): TYNetwork
vb	function **yFirstNetwork**( ) As YNetwork
cs	static YNetwork **FirstNetwork**( )
java	static YNetwork **FirstNetwork**( )
uwp	static YNetwork **FirstNetwork**( )
py	**FirstNetwork**( )
php	function **yFirstNetwork**( )
es	static **FirstNetwork**( )

Utiliser la fonction `YNetwork.nextNetwork()` pour itérer sur les autres interfaces réseau.

Retourne :

un pointeur sur un objet `YNetwork`, correspondant à la première interface réseau accessible en ligne, ou `null` si il n'y a pas de interfaces réseau disponibles.

YNetwork.FirstNetworkInContext() YNetwork
YNetwork.FirstNetworkInContext()

Commence l'énumération des interfaces réseau accessibles par la librairie.

java static YNetwork **FirstNetworkInContext**(YAPIContext **yctx**)
uwp static YNetwork **FirstNetworkInContext**(YAPIContext **yctx**)
es static **FirstNetworkInContext**(**yctx**)

Utiliser la fonction `YNetwork.nextNetwork()` pour itérer sur les autres interfaces réseau.

Paramètres :

yctx un contexte YAPI.

Retourne :

un pointeur sur un objet `YNetwork`, correspondant à la première interface réseau accessible en ligne, ou `null` si il n'y a pas de interfaces réseau disponibles.

YNetwork.GetSimilarFunctions() YNetwork
YNetwork.GetSimilarFunctions()

Enumère toutes les fonctions de type Network disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).

dnp static new string[] **GetSimilarFunctions**( )

Chaque chaîne retournée peut être passée en argument à la méthode `YNetwork.FindNetwork` pour obtenir une objet permettant d'intéragir avec le module correspondant.

Retourne :

un tableau de chaînes de caractères, contenant les identifiants matériels de chaque fonction disponible trouvée.

network→AdminPassword YNetwork

Chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide.

dnp string **AdminPassword**

Modifiable. Modifie le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module. Si la valeur fournie est une chaîne vide, plus aucun mot de passe n'est nécessaire. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→AdvertisedValue — YNetwork

Courte chaîne de caractères représentant l'état courant de la fonction.

dnp string **AdvertisedValue**

network→CallbackCredentials YNetwork

Version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide.

dnp string **CallbackCredentials**

Modifiable. Modifie le laisser-passer pour se connecter à l'adresse de callback. Le laisser-passer doit être fourni tel que retourné par la fonction `get_callbackCredentials`, sous la forme `username:hash`. La valeur du hash dépend de la méthode d'autorisation implémentée par le callback. Pour une autorisation de type Basic, le hash est le MD5 de la chaîne `username:password`. Pour une autorisation de type Digest, le hash est le MD5 de la chaîne `username:realm:password`. Pour une utilisation simplifiée, utilisez la fonction `callbackLogin`. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→CallbackEncoding YNetwork

Encodage à utiliser pour représenter les valeurs notifiées par callback.

dnp int **CallbackEncoding**

Valeurs possibles:

Y_CALLBACKENCODING_INVALID	= 0
Y_CALLBACKENCODING_FORM	= 1
Y_CALLBACKENCODING_JSON	= 2
Y_CALLBACKENCODING_JSON_ARRAY	= 3
Y_CALLBACKENCODING_CSV	= 4
Y_CALLBACKENCODING_YOCTO_API	= 5
Y_CALLBACKENCODING_JSON_NUM	= 6
Y_CALLBACKENCODING_EMONCMS	= 7
Y_CALLBACKENCODING_AZURE	= 8
Y_CALLBACKENCODING_INFLUXDB	= 9
Y_CALLBACKENCODING_MQTT	= 10
Y_CALLBACKENCODING_YOCTO_API_JZON	= 11
Y_CALLBACKENCODING_PRTG	= 12

Modifiable. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→CallbackInitialDelay YNetwork

Attente initiale avant la première notification par callback, en secondes.

dnp int **CallbackInitialDelay**

Modifiable. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→CallbackMaxDelay YNetwork

Attente entre deux callback HTTP lorsque rien n'est à signaler, en secondes.

dnp int **CallbackMaxDelay**

Modifiable. Modifie l'attente entre deux callback HTTP lorsque rien n'est à signaler. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→CallbackMethod YNetwork

Méthode HTTP à utiliser pour signaler les changements d'état par callback.

dnp int **CallbackMethod**

Valeurs possibles:

Y_CALLBACKMETHOD_INVALID = 0
Y_CALLBACKMETHOD_POST = 1
Y_CALLBACKMETHOD_GET = 2
Y_CALLBACKMETHOD_PUT = 3

Modifiable. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→CallbackMinDelay YNetwork

Attente minimale entre deux callbacks HTTP, en secondes.

dnp int **CallbackMinDelay**

Modifiable. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→CallbackSchedule YNetwork

Planification des callbacks HTTP, sous forme de chaîne de caractères.

dnp string **CallbackSchedule**

Modifiable. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→CallbackUrl YNetwork

Adresse (URL) de callback à notifier lors de changement d'état significatifs.

dnp string **CallbackUrl**

Modifiable. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→DefaultPage YNetwork

Page HTML à envoyer pour l'URL "/" **Modifiable**.

dnp string **DefaultPage**

Modifie la page HTML par défaut du hub. Si aucune valeur n'est attribuée le hub retourne index.html qui est l'interface web du hub. Il est possible de changer cet page pour un fichier qui a été uploadé sur le hub. Attention, la taille maximale permise pour le nom de fichier est de 15 caractères. Si vous changez ce paramètre, n'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→Discoverable — YNetwork

état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).

dnp int **Discoverable**

Valeurs possibles:

```
Y_DISCOVERABLE_INVALID = 0
Y_DISCOVERABLE_FALSE   = 1
Y_DISCOVERABLE_TRUE    = 2
```

Modifiable. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→FriendlyName — YNetwork

Identifiant global de la fonction au format NOM_MODULE.NOM_FONCTION.

dnp string **FriendlyName**

Le chaîne retournée utilise soit les noms logiques du module et de la fonction si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel de la fonction (par exemple: MyCustomName.relay1)

network→FunctionId — YNetwork

Identifiant matériel de l'interface réseau, sans référence au module.

dnp string **FunctionId**

Par example `relay1`.

network→HardwareId — YNetwork

Identifiant matériel unique de la fonction au format SERIAL.FUNCTIONID.

dnp string **HardwareId**

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de la fonction (par example RELAYLO1-123456.relay1).

network→HttpPort YNetwork

Port TCP utilisé pour l'interface Web du hub.

dnp int **HttpPort**

Modifiable. La valeur par défaut est le port 80, utilisé habituellement pat tous les serveurs web. Indépendamment de la valeur de ce paramètre, le hub répond toujours au port 4444, qui est utilisé par défaut par la librairie de programmation Yoctopuce. Si vous changez ce paramètre, n'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→IpAddress YNetwork

Adresse IP utilisée par le module Yoctopuce.

dnp string **IpAddress**

Il peut s'agir d'une adresse configurée statiquement, ou d'une adresse reçue par un serveur DHCP.

network→IsOnline YNetwork

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

dnp bool **IsOnline**

Si les valeurs des attributs en cache de la fonction sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

network→LogicalName YNetwork

Nom logique de la fonction.

dnp string **LogicalName**

Modifiable. Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→MacAddress YNetwork

Adresse MAC de l'interface réseau, unique pour chaque module.

dnp string **MacAddress**

L'adresse MAC est aussi présente sur un autocollant sur le module, représentée en chiffres et en code-barres.

network→NtpServer YNetwork

Adresse IP du serveur de NTP à utiliser pour maintenir le module à l'heure.

dnp string **NtpServer**

Modifiable. Modifie l'adresse IP du serveur NTP que le module doit utiliser. Utilisez une chaine vide pour restaurer l'adresse pré-configurée en usine. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

network→PrimaryDNS YNetwork

Adresse IP du serveur de noms primaire que le module doit utiliser.

dnp string **PrimaryDNS**

Modifiable. En mode DHCP, si une valeur est spécifiée, elle remplacera celle reçue du serveur DHCP. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

network→Readiness YNetwork

état de fonctionnement atteint par l'interface réseau.

dnp int **Readiness**

Le niveau zéro (DOWN_0) signifie qu'aucun support réseau matériel n'a été détecté. Soit il n'y a pas de signal sur le cable réseau, soit le point d'accès sans fil choisi n'est pas détecté. Le niveau 1 (LIVE_1) est atteint lorsque le réseau est détecté, mais n'est pas encore connecté. Pour un réseau sans fil, cela confirme la l'existence du SSID configuré. Le niveau 2 (LINK_2) est atteint lorsque le support matériel du réseau est fonctionnel. Pour une connection réseau filaire, le niveau 2 signifie que le cable est connecté aux deux bouts. Pour une connection à un point d'accès réseau sans fil, il démontre que les paramètres de sécurités configurés sont corrects. Pour une connection sans fil en mode ad-hoc, cela signifie qu'il y a au moins un partenaire sur le réseau ad-hoc. Le niveau 3 (DHCP_3) est atteint lorsque qu'une adresse IP a été obtenue par DHCP. Le niveau 4 (DNS_4) est atteint lorsqu'un serveur DNS est joignable par le réseau. Le niveau 5 (WWW_5) est atteint lorsque la connectivité globale à internet est avérée par l'obtention de l'heure courante sur une serveur NTP.

Valeurs possibles:

```
Y_READINESS_INVALID = 0
Y_READINESS_DOWN    = 1
Y_READINESS_EXISTS  = 2
Y_READINESS_LINKED  = 3
Y_READINESS_LAN_OK  = 4
Y_READINESS_WWW_OK  = 5
```

network→SecondaryDNS YNetwork

Adresse IP du serveur de noms secondaire que le module doit utiliser.

dnp string **SecondaryDNS**

Modifiable. Modifie l'adresse IP du serveur de nom secondaire que le module doit utiliser. En mode DHCP, si une valeur est spécifiée, elle remplacera celle reçue du serveur DHCP. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

network→SerialNumber YNetwork

Numéro de série du module, préprogrammé en usine.

dnp string **SerialNumber**

network→UserPassword YNetwork

Chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide.

dnp string **UserPassword**

Modifiable. Modifie le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module. Si la valeur fournie est une chaîne vide, plus aucun mot de passe n'est nécessaire. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→WwwWatchdogDelay YNetwork

Durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

dnp int **WwwWatchdogDelay**

Une valeur nulle désactive le redémarrage automatique en cas de perte de connectivité WWW.

Modifiable. Une valeur nulle désactive le redémarrage automatique en cas de perte de connectivité WWW. La plus petite durée non-nulle utilisable est 90 secondes. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

network→callbackLogin() YNetwork

Contacte le callback de notification et sauvegarde un laisser-passer pour s'y connecter.

```
js    function callbackLogin( username, password)
cpp   int callbackLogin( string username, string password)
m     -(int) callbackLogin : (NSString*) username
                           : (NSString*) password
pas   integer callbackLogin( username: string, password: string): integer
vb    function callbackLogin( ByVal username As String,
                              ByVal password As String) As Integer
cs    int callbackLogin( string username, string password)
dnp   int callbackLogin( string username, string password)
java  int callbackLogin( String username, String password)
py    callbackLogin( username, password)
php   function callbackLogin( $username, $password)
es    async callbackLogin( username, password)
cmd   YNetwork target callbackLogin username password
```

Le mot de passe ne sera pas stocké dans le module, mais seulement une version hashée non réversible. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

username nom d'utilisateur pour s'identifier au callback

password mot de passe pour s'identifier au callback

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→clearCache() YNetwork

Invalide le cache.

js	function **clearCache**( )
cpp	void **clearCache**( )
m	-(void) **clearCache**
pas	**clearCache**( )
vb	procedure **clearCache**( )
cs	void **clearCache**( )
java	void **clearCache**( )
py	**clearCache**( )
php	function **clearCache**( )
es	async **clearCache**( )

Invalide le cache des valeurs courantes de l'interface réseau. Force le prochain appel à une méthode get_xxx() ou loadxxx() pour charger les les données depuis le module.

network→describe() YNetwork

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'interface réseau au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     string describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      describe( )
php     function describe( )
es      async describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

Retourne :
une chaîne de caractères décrivant l'interface réseau (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

network→get_adminPassword() YNetwork
network→adminPassword()

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide.

```
js      function get_adminPassword( )
cpp     string get_adminPassword( )
m       -(NSString*) adminPassword
pas     string get_adminPassword( ): string
vb      function get_adminPassword( ) As String
cs      string get_adminPassword( )
dnp     string get_adminPassword( )
java    String get_adminPassword( )
uwp     async Task<string> get_adminPassword( )
py      get_adminPassword( )
php     function get_adminPassword( )
es      async get_adminPassword( )
cmd     YNetwork target get_adminPassword
```

Retourne :

une chaîne de caractères représentant une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "admin", ou sinon une chaîne vide

En cas d'erreur, déclenche une exception ou retourne `Y_ADMINPASSWORD_INVALID`.

network→get_advertisedValue() YNetwork
network→advertisedValue()

Retourne la valeur courante de l'interface réseau (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     string get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
dnp     string get_advertisedValue( )
java    String get_advertisedValue( )
uwp     async Task<string> get_advertisedValue( )
py      get_advertisedValue( )
php     function get_advertisedValue( )
es      async get_advertisedValue( )
cmd     YNetwork target get_advertisedValue
```

Retourne :
une chaîne de caractères représentant la valeur courante de l'interface réseau (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

network→get_callbackCredentials() YNetwork
network→callbackCredentials()

Retourne une version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide.

```
js      function get_callbackCredentials( )
cpp     string get_callbackCredentials( )
m       -(NSString*) callbackCredentials
pas     string get_callbackCredentials( ): string
vb      function get_callbackCredentials( ) As String
cs      string get_callbackCredentials( )
dnp     string get_callbackCredentials( )
java    String get_callbackCredentials( )
uwp     async Task<string> get_callbackCredentials( )
py      get_callbackCredentials( )
php     function get_callbackCredentials( )
es      async get_callbackCredentials( )
cmd     YNetwork target get_callbackCredentials
```

Retourne :

une chaîne de caractères représentant une version hashée du laisser-passer pour le callback de notification s'il a été configuré, ou sinon une chaîne vide

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKCREDENTIALS_INVALID`.

network→get_callbackEncoding() YNetwork
network→callbackEncoding()

Retourne l'encodage à utiliser pour représenter les valeurs notifiées par callback.

js	function **get_callbackEncoding**( )
cpp	Y_CALLBACKENCODING_enum **get_callbackEncoding**( )
m	-(Y_CALLBACKENCODING_enum) callbackEncoding
pas	Integer **get_callbackEncoding**( ): Integer
vb	function **get_callbackEncoding**( ) As Integer
cs	int **get_callbackEncoding**( )
dnp	int **get_callbackEncoding**( )
java	int **get_callbackEncoding**( )
uwp	async Task<int> **get_callbackEncoding**( )
py	**get_callbackEncoding**( )
php	function **get_callbackEncoding**( )
es	async **get_callbackEncoding**( )
cmd	YNetwork **target get_callbackEncoding**

Retourne :

une valeur parmi `Y_CALLBACKENCODING_FORM`, `Y_CALLBACKENCODING_JSON`, `Y_CALLBACKENCODING_JSON_ARRAY`, `Y_CALLBACKENCODING_CSV`, `Y_CALLBACKENCODING_YOCTO_API`, `Y_CALLBACKENCODING_JSON_NUM`, `Y_CALLBACKENCODING_EMONCMS`, `Y_CALLBACKENCODING_AZURE`, `Y_CALLBACKENCODING_INFLUXDB`, `Y_CALLBACKENCODING_MQTT`, `Y_CALLBACKENCODING_YOCTO_API_JZON` et `Y_CALLBACKENCODING_PRTG` représentant l'encodage à utiliser pour représenter les valeurs notifiées par callback

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKENCODING_INVALID`.

network→get_callbackInitialDelay()
network→callbackInitialDelay()

YNetwork

Retourne l'attente initiale avant la première notification par callback, en secondes.

```
js      function get_callbackInitialDelay( )
cpp     int get_callbackInitialDelay( )
m       -(int) callbackInitialDelay
pas     LongInt get_callbackInitialDelay( ): LongInt
vb      function get_callbackInitialDelay( ) As Integer
cs      int get_callbackInitialDelay( )
dnp     int get_callbackInitialDelay( )
java    int get_callbackInitialDelay( )
uwp     async Task<int> get_callbackInitialDelay( )
py      get_callbackInitialDelay( )
php     function get_callbackInitialDelay( )
es      async get_callbackInitialDelay( )
cmd     YNetwork target get_callbackInitialDelay
```

Retourne :

un entier représentant l'attente initiale avant la première notification par callback, en secondes

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKINITIALDELAY_INVALID`.

network→get_callbackMaxDelay() YNetwork
network→callbackMaxDelay()

Retourne l'attente entre deux callback HTTP lorsque rien n'est à signaler, en secondes.

js	function **get_callbackMaxDelay**( )
cpp	int **get_callbackMaxDelay**( )
m	-(int) callbackMaxDelay
pas	LongInt **get_callbackMaxDelay**( ): LongInt
vb	function **get_callbackMaxDelay**( ) As Integer
cs	int **get_callbackMaxDelay**( )
dnp	int **get_callbackMaxDelay**( )
java	int **get_callbackMaxDelay**( )
uwp	async Task<int> **get_callbackMaxDelay**( )
py	**get_callbackMaxDelay**( )
php	function **get_callbackMaxDelay**( )
es	async **get_callbackMaxDelay**( )
cmd	YNetwork **target get_callbackMaxDelay**

Retourne :
un entier représentant l'attente entre deux callback HTTP lorsque rien n'est à signaler, en secondes

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKMAXDELAY_INVALID`.

network→get_callbackMethod() YNetwork
network→callbackMethod()

Retourne la méthode HTTP à utiliser pour signaler les changements d'état par callback.

```
js      function get_callbackMethod( )
cpp     Y_CALLBACKMETHOD_enum get_callbackMethod( )
m       -(Y_CALLBACKMETHOD_enum) callbackMethod
pas     Integer get_callbackMethod( ): Integer
vb      function get_callbackMethod( ) As Integer
cs      int get_callbackMethod( )
dnp     int get_callbackMethod( )
java    int get_callbackMethod( )
uwp     async Task<int> get_callbackMethod( )
py      get_callbackMethod( )
php     function get_callbackMethod( )
es      async get_callbackMethod( )
cmd     YNetwork target get_callbackMethod
```

Retourne :

une valeur parmi `Y_CALLBACKMETHOD_POST`, `Y_CALLBACKMETHOD_GET` et `Y_CALLBACKMETHOD_PUT` représentant la méthode HTTP à utiliser pour signaler les changements d'état par callback

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKMETHOD_INVALID`.

network→get_callbackMinDelay() YNetwork
network→callbackMinDelay()

Retourne l'attente minimale entre deux callbacks HTTP, en secondes.

```
js      function get_callbackMinDelay( )
cpp     int get_callbackMinDelay( )
m       -(int) callbackMinDelay
pas     LongInt get_callbackMinDelay( ): LongInt
vb      function get_callbackMinDelay( ) As Integer
cs      int get_callbackMinDelay( )
dnp     int get_callbackMinDelay( )
java    int get_callbackMinDelay( )
uwp     async Task<int> get_callbackMinDelay( )
py      get_callbackMinDelay( )
php     function get_callbackMinDelay( )
es      async get_callbackMinDelay( )
cmd     YNetwork target get_callbackMinDelay
```

Retourne :
un entier représentant l'attente minimale entre deux callbacks HTTP, en secondes

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKMINDELAY_INVALID`.

network→get_callbackSchedule() YNetwork
network→callbackSchedule()

Retourne la planification des callbacks HTTP, sous forme de chaîne de caractères.

```
js      function get_callbackSchedule( )
cpp     string get_callbackSchedule( )
m       -(NSString*) callbackSchedule
pas     string get_callbackSchedule( ): string
vb      function get_callbackSchedule( ) As String
cs      string get_callbackSchedule( )
dnp     string get_callbackSchedule( )
java    String get_callbackSchedule( )
uwp     async Task<string> get_callbackSchedule( )
py      get_callbackSchedule( )
php     function get_callbackSchedule( )
es      async get_callbackSchedule( )
cmd     YNetwork target get_callbackSchedule
```

Retourne :
une chaîne de caractères représentant la planification des callbacks HTTP, sous forme de chaîne de caractères

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKSCHEDULE_INVALID`.

network→get_callbackUrl() YNetwork
network→callbackUrl()

Retourne l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

```
js      function get_callbackUrl( )
cpp     string get_callbackUrl( )
m       -(NSString*) callbackUrl
pas     string get_callbackUrl( ): string
vb      function get_callbackUrl( ) As String
cs      string get_callbackUrl( )
dnp     string get_callbackUrl( )
java    String get_callbackUrl( )
uwp     async Task<string> get_callbackUrl( )
py      get_callbackUrl( )
php     function get_callbackUrl( )
es      async get_callbackUrl( )
cmd     YNetwork target get_callbackUrl
```

Retourne :

une chaîne de caractères représentant l'adresse (URL) de callback à notifier lors de changement d'état significatifs

En cas d'erreur, déclenche une exception ou retourne `Y_CALLBACKURL_INVALID`.

network→get_defaultPage()
network→defaultPage()

YNetwork

Retourne la page HTML à envoyer pour l'URL "/"

```
js      function get_defaultPage( )
cpp     string get_defaultPage( )
m       -(NSString*) defaultPage
pas     string get_defaultPage( ): string
vb      function get_defaultPage( ) As String
cs      string get_defaultPage( )
dnp     string get_defaultPage( )
java    String get_defaultPage( )
uwp     async Task<string> get_defaultPage( )
py      get_defaultPage( )
php     function get_defaultPage( )
es      async get_defaultPage( )
cmd     YNetwork target get_defaultPage
```

Retourne :
une chaîne de caractères représentant la page HTML à envoyer pour l'URL "/"

En cas d'erreur, déclenche une exception ou retourne `Y_DEFAULTPAGE_INVALID`.

network→get_discoverable()
network→discoverable()

YNetwork

Retourne l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).

js	function **get_discoverable**( )
cpp	Y_DISCOVERABLE_enum **get_discoverable**( )
m	-(Y_DISCOVERABLE_enum) discoverable
pas	Integer **get_discoverable**( ): Integer
vb	function **get_discoverable**( ) As Integer
cs	int **get_discoverable**( )
dnp	int **get_discoverable**( )
java	int **get_discoverable**( )
uwp	async Task<int> **get_discoverable**( )
py	**get_discoverable**( )
php	function **get_discoverable**( )
es	async **get_discoverable**( )
cmd	YNetwork **target** **get_discoverable**

Retourne :

soit `Y_DISCOVERABLE_FALSE`, soit `Y_DISCOVERABLE_TRUE`, selon l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour)

En cas d'erreur, déclenche une exception ou retourne `Y_DISCOVERABLE_INVALID`.

network→get_errorMessage() YNetwork
network→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

js	function **get_errorMessage**( )
cpp	string **get_errorMessage**( )
m	-(NSString*) errorMessage
pas	string **get_errorMessage**( ): string
vb	function **get_errorMessage**( ) As String
cs	string **get_errorMessage**( )
java	String **get_errorMessage**( )
py	**get_errorMessage**( )
php	function **get_errorMessage**( )
es	**get_errorMessage**( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'interface réseau.

network→get_errorType() YNetwork
network→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'interface réseau.

js	function **get_errorType**( )
cpp	YRETCODE **get_errorType**( )
m	-(YRETCODE) errorType
pas	YRETCODE **get_errorType**( ): YRETCODE
vb	function **get_errorType**( ) As YRETCODE
cs	YRETCODE **get_errorType**( )
java	int **get_errorType**( )
py	**get_errorType**( )
php	function **get_errorType**( )
es	**get_errorType**( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'interface réseau.

network→get_friendlyName()
network→friendlyName()

YNetwork

Retourne un identifiant global de l'interface réseau au format NOM_MODULE.NOM_FONCTION.

```
js      function get_friendlyName( )
cpp     string get_friendlyName( )
m       -(NSString*) friendlyName
cs      string get_friendlyName( )
dnp     string get_friendlyName( )
java    String get_friendlyName( )
py      get_friendlyName( )
php     function get_friendlyName( )
es      async get_friendlyName( )
```

Le chaîne retournée utilise soit les noms logiques du module et de l'interface réseau si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel de l'interface réseau (par exemple: MyCustomName.relay1)

Retourne :

une chaîne de caractères identifiant l'interface réseau en utilisant les noms logiques (ex: MyCustomName.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

network→get_functionDescriptor() YNetwork
network→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function **get_functionDescriptor**( )
cpp	YFUN_DESCR **get_functionDescriptor**( )
m	-(YFUN_DESCR) functionDescriptor
pas	YFUN_DESCR **get_functionDescriptor**( ): YFUN_DESCR
vb	function **get_functionDescriptor**( ) As YFUN_DESCR
cs	YFUN_DESCR **get_functionDescriptor**( )
java	String **get_functionDescriptor**( )
py	**get_functionDescriptor**( )
php	function **get_functionDescriptor**( )
es	async **get_functionDescriptor**( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

Retourne :
un identifiant de type YFUN_DESCR.

Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

network→get_functionId()
network→functionId()

YNetwork

Retourne l'identifiant matériel de l'interface réseau, sans référence au module.

```
js      function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
dnp     string get_functionId( )
java    String get_functionId( )
py      get_functionId( )
php     function get_functionId( )
es      async get_functionId( )
```

Par example `relay1`.

Retourne :
une chaîne de caractères identifiant l'interface réseau (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

network→get_hardwareId() YNetwork
network→hardwareId()

Retourne l'identifiant matériel unique de l'interface réseau au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
dnp     string get_hardwareId( )
java    String get_hardwareId( )
py      get_hardwareId( )
php     function get_hardwareId( )
es      async get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'interface réseau (par example RELAYLO1-123456.relay1).

Retourne :
une chaîne de caractères identifiant l'interface réseau (ex: RELAYLO1-123456.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

network→get_httpPort()
network→httpPort()

YNetwork

Retourne le port TCP utilisé pour l'interface Web du hub.

```
js      function get_httpPort( )
cpp     int get_httpPort( )
m       -(int) httpPort
pas     LongInt get_httpPort( ): LongInt
vb      function get_httpPort( ) As Integer
cs      int get_httpPort( )
dnp     int get_httpPort( )
java    int get_httpPort( )
uwp     async Task<int> get_httpPort( )
py      get_httpPort( )
php     function get_httpPort( )
es      async get_httpPort( )
cmd     YNetwork target get_httpPort
```

Retourne :
un entier représentant le port TCP utilisé pour l'interface Web du hub

En cas d'erreur, déclenche une exception ou retourne `Y_HTTPPORT_INVALID`.

network→get_ipAddress()
network→ipAddress()

YNetwork

Retourne l'adresse IP utilisée par le module Yoctopuce.

js	function **get_ipAddress**( )
cpp	string **get_ipAddress**( )
m	-(NSString*) ipAddress
pas	string **get_ipAddress**( ): string
vb	function **get_ipAddress**( ) As String
cs	string **get_ipAddress**( )
dnp	string **get_ipAddress**( )
java	String **get_ipAddress**( )
uwp	async Task<string> **get_ipAddress**( )
py	**get_ipAddress**( )
php	function **get_ipAddress**( )
es	async **get_ipAddress**( )
cmd	YNetwork **target** **get_ipAddress**

Il peut s'agir d'une adresse configurée statiquement, ou d'une adresse reçue par un serveur DHCP.

Retourne :
une chaîne de caractères représentant l'adresse IP utilisée par le module Yoctopuce

En cas d'erreur, déclenche une exception ou retourne `Y_IPADDRESS_INVALID`.

network→get_ipConfig()
network→ipConfig()

YNetwork

Retourne la configuration IP de l'interface réseau.

```
js      function get_ipConfig( )
cpp     string get_ipConfig( )
m       -(NSString*) ipConfig
pas     string get_ipConfig( ): string
vb      function get_ipConfig( ) As String
cs      string get_ipConfig( )
dnp     string get_ipConfig( )
java    String get_ipConfig( )
uwp     async Task<string> get_ipConfig( )
py      get_ipConfig( )
php     function get_ipConfig( )
es      async get_ipConfig( )
cmd     YNetwork target get_ipConfig
```

Si l'interface réseau est configurée pour utiliser une adresse IP assignée manuellement (adresse IP statique) la chaîne commence par "STATIC:" et est suivie pas l'adresse IP, la longueur du masque de sous-réseau et l'adresse IP de la passerelle. Ces trois paramètres sont séparés par le caractère "/". Par exemple: "STATIC:192.168.1.14/16/192.168.1.1"

Si l'interface réseau est configurée pour utiliser une adresse assignée automatiquement par DHCP la chaîne commence par "DHCP:" et est suivie d'une adresse IP, d'une longueur du masque de sous-réseau et d'une adresse IP de passerelle. Ces trois paramètres sont séparés par le caractère "/" et sont utilisés si aucun serveur DHCP ne répond.

Retourne :
une chaîne de caractères représentant la configuration IP de l'interface réseau

En cas d'erreur, déclenche une exception ou retourne `Y_IPCONFIG_INVALID`.

network→get_logicalName() YNetwork
network→logicalName()

Retourne le nom logique de l'interface réseau.

```
js      function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     string get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
dnp     string get_logicalName( )
java    String get_logicalName( )
uwp     async Task<string> get_logicalName( )
py      get_logicalName( )
php     function get_logicalName( )
es      async get_logicalName( )
cmd     YNetwork target get_logicalName
```

Retourne :
une chaîne de caractères représentant le nom logique de l'interface réseau.

En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

network→get_macAddress()
network→macAddress()

YNetwork

Retourne l'adresse MAC de l'interface réseau, unique pour chaque module.

```
js    function get_macAddress( )
cpp   string get_macAddress( )
m     -(NSString*) macAddress
pas   string get_macAddress( ): string
vb    function get_macAddress( ) As String
cs    string get_macAddress( )
dnp   string get_macAddress( )
java  String get_macAddress( )
uwp   async Task<string> get_macAddress( )
py    get_macAddress( )
php   function get_macAddress( )
es    async get_macAddress( )
cmd   YNetwork target get_macAddress
```

L'adresse MAC est aussi présente sur un autocollant sur le module, représentée en chiffres et en code-barres.

Retourne :
une chaîne de caractères représentant l'adresse MAC de l'interface réseau, unique pour chaque module

En cas d'erreur, déclenche une exception ou retourne `Y_MACADDRESS_INVALID`.

network→get_module()
network→module()

YNetwork

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function **get_module**( )
cpp	YModule * **get_module**( )
m	-(YModule*) module
pas	TYModule **get_module**( ): TYModule
vb	function **get_module**( ) As YModule
cs	YModule **get_module**( )
dnp	YModuleProxy **get_module**( )
java	YModule **get_module**( )
py	**get_module**( )
php	function **get_module**( )
es	async **get_module**( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Retourne :
une instance de YModule

network→get_module_async() YNetwork
network→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

network→get_ntpServer() YNetwork
network→ntpServer()

Retourne l'adresse IP du serveur de NTP à utiliser pour maintenir le module à l'heure.

```
js      function get_ntpServer( )
cpp     string get_ntpServer( )
m       -(NSString*) ntpServer
pas     string get_ntpServer( ): string
vb      function get_ntpServer( ) As String
cs      string get_ntpServer( )
dnp     string get_ntpServer( )
java    String get_ntpServer( )
uwp     async Task<string> get_ntpServer( )
py      get_ntpServer( )
php     function get_ntpServer( )
es      async get_ntpServer( )
cmd     YNetwork target get_ntpServer
```

Retourne :

une chaîne de caractères représentant l'adresse IP du serveur de NTP à utiliser pour maintenir le module à l'heure

En cas d'erreur, déclenche une exception ou retourne `Y_NTPSERVER_INVALID`.

network→get_poeCurrent() YNetwork
network→poeCurrent()

Retourne le courant consommé par le module depuis Power-over-Ethernet (PoE), en milliampères.

```
js      function get_poeCurrent( )
cpp     int get_poeCurrent( )
m       -(int) poeCurrent
pas     LongInt get_poeCurrent( ): LongInt
vb      function get_poeCurrent( ) As Integer
cs      int get_poeCurrent( )
dnp     int get_poeCurrent( )
java    int get_poeCurrent( )
uwp     async Task<int> get_poeCurrent( )
py      get_poeCurrent( )
php     function get_poeCurrent( )
es      async get_poeCurrent( )
cmd     YNetwork target get_poeCurrent
```

La consommation est mesurée après conversion en 5 Volt, et ne doit jamais dépasser 1800 mA.

Retourne :
un entier représentant le courant consommé par le module depuis Power-over-Ethernet (PoE), en milliampères

En cas d'erreur, déclenche une exception ou retourne `Y_POECURRENT_INVALID`.

network→get_primaryDNS() YNetwork
network→primaryDNS()

Retourne l'adresse IP du serveur de noms primaire que le module doit utiliser.

js	function **get_primaryDNS**( )
cpp	string **get_primaryDNS**( )
m	-(NSString*) primaryDNS
pas	string **get_primaryDNS**( ): string
vb	function **get_primaryDNS**( ) As String
cs	string **get_primaryDNS**( )
dnp	string **get_primaryDNS**( )
java	String **get_primaryDNS**( )
uwp	async Task<string> **get_primaryDNS**( )
py	**get_primaryDNS**( )
php	function **get_primaryDNS**( )
es	async **get_primaryDNS**( )
cmd	YNetwork **target get_primaryDNS**

Retourne :
une chaîne de caractères représentant l'adresse IP du serveur de noms primaire que le module doit utiliser

En cas d'erreur, déclenche une exception ou retourne `Y_PRIMARYDNS_INVALID`.

network→get_readiness()
network→readiness()

YNetwork

Retourne l'état de fonctionnement atteint par l'interface réseau.

```
js      function get_readiness( )
cpp     Y_READINESS_enum get_readiness( )
m       -(Y_READINESS_enum) readiness
pas     Integer get_readiness( ): Integer
vb      function get_readiness( ) As Integer
cs      int get_readiness( )
dnp     int get_readiness( )
java    int get_readiness( )
uwp     async Task<int> get_readiness( )
py      get_readiness( )
php     function get_readiness( )
es      async get_readiness( )
cmd     YNetwork target get_readiness
```

Le niveau zéro (DOWN_0) signifie qu'aucun support réseau matériel n'a été détecté. Soit il n'y a pas de signal sur le cable réseau, soit le point d'accès sans fil choisi n'est pas détecté. Le niveau 1 (LIVE_1) est atteint lorsque le réseau est détecté, mais n'est pas encore connecté. Pour un réseau sans fil, cela confirme la l'existence du SSID configuré. Le niveau 2 (LINK_2) est atteint lorsque le support matériel du réseau est fonctionnel. Pour une connection réseau filaire, le niveau 2 signifie que le cable est connecté aux deux bouts. Pour une connection à un point d'accès réseau sans fil, il démontre que les paramètres de sécurités configurés sont corrects. Pour une connection sans fil en mode ad-hoc, cela signifie qu'il y a au moins un partenaire sur le réseau ad-hoc. Le niveau 3 (DHCP_3) est atteint lorsque qu'une adresse IP a été obtenue par DHCP. Le niveau 4 (DNS_4) est atteint lorsqu'un serveur DNS est joignable par le réseau. Le niveau 5 (WWW_5) est atteint lorsque la connectivité globale à internet est avérée par l'obtention de l'heure courante sur une serveur NTP.

Retourne :

une valeur parmi `Y_READINESS_DOWN`, `Y_READINESS_EXISTS`, `Y_READINESS_LINKED`, `Y_READINESS_LAN_OK` et `Y_READINESS_WWW_OK` représentant l'état de fonctionnement atteint par l'interface réseau

En cas d'erreur, déclenche une exception ou retourne `Y_READINESS_INVALID`.

network→get_router() YNetwork
network→router()

Retourne l'adresse IP du routeur (passerelle) utilisé par le module (*default gateway*).

```
js      function get_router( )
cpp     string get_router( )
m       -(NSString*) router
pas     string get_router( ): string
vb      function get_router( ) As String
cs      string get_router( )
dnp     string get_router( )
java    String get_router( )
uwp     async Task<string> get_router( )
py      get_router( )
php     function get_router( )
es      async get_router( )
cmd     YNetwork target get_router
```

Retourne :

une chaîne de caractères représentant l'adresse IP du routeur (passerelle) utilisé par le module (*default gateway*)

En cas d'erreur, déclenche une exception ou retourne `Y_ROUTER_INVALID`.

network→get_secondaryDNS()
network→secondaryDNS()

YNetwork

Retourne l'adresse IP du serveur de noms secondaire que le module doit utiliser.

```
js      function get_secondaryDNS( )
cpp     string get_secondaryDNS( )
m       -(NSString*) secondaryDNS
pas     string get_secondaryDNS( ): string
vb      function get_secondaryDNS( ) As String
cs      string get_secondaryDNS( )
dnp     string get_secondaryDNS( )
java    String get_secondaryDNS( )
uwp     async Task<string> get_secondaryDNS( )
py      get_secondaryDNS( )
php     function get_secondaryDNS( )
es      async get_secondaryDNS( )
cmd     YNetwork target get_secondaryDNS
```

Retourne :

une chaîne de caractères représentant l'adresse IP du serveur de noms secondaire que le module doit utiliser

En cas d'erreur, déclenche une exception ou retourne `Y_SECONDARYDNS_INVALID`.

network→get_serialNumber() YNetwork
network→serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

```
js      function get_serialNumber( )
cpp     string get_serialNumber( )
m       -(NSString*) serialNumber
pas     string get_serialNumber( ): string
vb      function get_serialNumber( ) As String
cs      string get_serialNumber( )
dnp     string get_serialNumber( )
java    String get_serialNumber( )
uwp     async Task<string> get_serialNumber( )
py      get_serialNumber( )
php     function get_serialNumber( )
es      async get_serialNumber( )
cmd     YNetwork target get_serialNumber
```

Retourne :

: une chaîne de caractères représentant le numéro de série du module, préprogrammé en usine.

En cas d'erreur, déclenche une exception ou retourne YModule.SERIALNUMBER_INVALID.

network→get_subnetMask()
network→subnetMask()

YNetwork

Retourne le masque de sous-réseau utilisé par le module.

```
js      function get_subnetMask( )
cpp     string get_subnetMask( )
m       -(NSString*) subnetMask
pas     string get_subnetMask( ): string
vb      function get_subnetMask( ) As String
cs      string get_subnetMask( )
dnp     string get_subnetMask( )
java    String get_subnetMask( )
uwp     async Task<string> get_subnetMask( )
py      get_subnetMask( )
php     function get_subnetMask( )
es      async get_subnetMask( )
cmd     YNetwork target get_subnetMask
```

Retourne :
une chaîne de caractères représentant le masque de sous-réseau utilisé par le module

En cas d'erreur, déclenche une exception ou retourne `Y_SUBNETMASK_INVALID`.

network→get_userData() YNetwork
network→userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function **get_userData**( )
cpp	void * **get_userData**( )
m	-(id) userData
pas	Tobject **get_userData**( ): Tobject
vb	function **get_userData**( ) As Object
cs	object **get_userData**( )
java	Object **get_userData**( )
py	**get_userData**( )
php	function **get_userData**( )
es	async **get_userData**( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Retourne :
l'objet stocké précédemment par l'appelant.

network→get_userPassword() YNetwork
network→userPassword()

Retourne une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide.

```
js      function get_userPassword( )
cpp     string get_userPassword( )
m       -(NSString*) userPassword
pas     string get_userPassword( ): string
vb      function get_userPassword( ) As String
cs      string get_userPassword( )
dnp     string get_userPassword( )
java    String get_userPassword( )
uwp     async Task<string> get_userPassword( )
py      get_userPassword( )
php     function get_userPassword( )
es      async get_userPassword( )
cmd     YNetwork target get_userPassword
```

Retourne :

une chaîne de caractères représentant une chaîne de hash si un mot de passe a été configuré pour l'utilisateur "user", ou sinon une chaîne vide

En cas d'erreur, déclenche une exception ou retourne `Y_USERPASSWORD_INVALID`.

network→get_wwwWatchdogDelay() YNetwork
network→wwwWatchdogDelay()

Retourne la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

js	function **get_wwwWatchdogDelay**( )
cpp	int **get_wwwWatchdogDelay**( )
m	-(int) wwwWatchdogDelay
pas	LongInt **get_wwwWatchdogDelay**( ): LongInt
vb	function **get_wwwWatchdogDelay**( ) As Integer
cs	int **get_wwwWatchdogDelay**( )
dnp	int **get_wwwWatchdogDelay**( )
java	int **get_wwwWatchdogDelay**( )
uwp	async Task<int> **get_wwwWatchdogDelay**( )
py	**get_wwwWatchdogDelay**( )
php	function **get_wwwWatchdogDelay**( )
es	async **get_wwwWatchdogDelay**( )
cmd	YNetwork **target** **get_wwwWatchdogDelay**

Une valeur nulle désactive le redémarrage automatique en cas de perte de connectivité WWW.

Retourne :
un entier représentant la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet

En cas d'erreur, déclenche une exception ou retourne `Y_WWWWATCHDOGDELAY_INVALID`.

network→isOnline() YNetwork

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

```
js    function isOnline( )
cpp   bool isOnline( )
m     -(BOOL) isOnline
pas   boolean isOnline( ): boolean
vb    function isOnline( ) As Boolean
cs    bool isOnline( )
dnp   bool isOnline( )
java  boolean isOnline( )
py    isOnline( )
php   function isOnline( )
es    async isOnline( )
```

Si les valeurs des attributs en cache de l'interface réseau sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Retourne :
`true` si l'interface réseau est joignable, `false` sinon

network→isOnline_async() YNetwork

Vérifie si le module hébergeant l'interface réseau est joignable, sans déclencher d'erreur.

```
js  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'interface réseau sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

network→isReadOnly() YNetwork

Test si la fonction est en lecture seule.

cpp	bool **isReadOnly**( )
m	-(bool) **isReadOnly**
pas	boolean **isReadOnly**( ): boolean
vb	function **isReadOnly**( ) As Boolean
cs	bool **isReadOnly**( )
dnp	bool **isReadOnly**( )
java	boolean **isReadOnly**( )
uwp	async Task<bool> **isReadOnly**( )
py	**isReadOnly**( )
php	function **isReadOnly**( )
es	async **isReadOnly**( )
cmd	YNetwork **target isReadOnly**

Retourne vrais si la fonction est protégé en ecriture ou que la fontion n'est pas disponible.

Retourne :
`true` si la fonction est protégé en ecriture ou que la fontion n'est pas disponible

network→load() YNetwork

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

```
js      function load( msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (u64) msValidity
pas     YRETCODE load( msValidity: u64): YRETCODE
vb      function load( ByVal msValidity As Long) As YRETCODE
cs      YRETCODE load( ulong msValidity)
java    int load( long msValidity)
py      load( msValidity)
php     function load( $msValidity)
es      async load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→loadAttribute() YNetwork

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

```
js      function loadAttribute( attrName)
cpp     string loadAttribute( string attrName)
m       -(NSString*) loadAttribute : (NSString*) attrName
pas     string loadAttribute( attrName: string): string
vb      function loadAttribute( ) As String
cs      string loadAttribute( string attrName)
dnp     string loadAttribute( string attrName)
java    String loadAttribute( String attrName)
uwp     async Task<string> loadAttribute( string attrName)
py      loadAttribute( attrName)
php     function loadAttribute( $attrName)
es      async loadAttribute( attrName)
```

Paramètres :

attrName le nom de l'attribut désiré

Retourne :

une chaîne de caractères représentant la valeur actuelle de l'attribut.

En cas d'erreur, déclenche une exception ou retourne un chaîne vide.

network→load_async() YNetwork

Met en cache les valeurs courantes de l'interface réseau, avec une durée de validité spécifiée.

```
js  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

msValidity	un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
callback	fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)
context	contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

network→muteValueCallbacks() YNetwork

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

```
js    function muteValueCallbacks( )
cpp   int muteValueCallbacks( )
m     -(int) muteValueCallbacks
pas   LongInt muteValueCallbacks( ): LongInt
vb    function muteValueCallbacks( ) As Integer
cs    int muteValueCallbacks( )
dnp   int muteValueCallbacks( )
java  int muteValueCallbacks( )
uwp   async Task<int> muteValueCallbacks( )
py    muteValueCallbacks( )
php   function muteValueCallbacks( )
es    async muteValueCallbacks( )
cmd   YNetwork target muteValueCallbacks
```

Vous pouvez utiliser cette fonction pour économiser la bande passante et le CPU sur les machines de faible puissance, ou pour éviter le déclanchement de callbacks HTTP. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→nextNetwork() YNetwork

Continue l'énumération des interfaces réseau commencée à l'aide de `yFirstNetwork()`. Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les interfaces réseau sont retournés.

```
js      function nextNetwork( )
cpp     YNetwork * nextNetwork( )
m       -(YNetwork*) nextNetwork
pas     TYNetwork nextNetwork( ): TYNetwork
vb      function nextNetwork( ) As YNetwork
cs      YNetwork nextNetwork( )
java    YNetwork nextNetwork( )
uwp     YNetwork nextNetwork( )
py      nextNetwork( )
php     function nextNetwork( )
es      nextNetwork( )
```

Si vous souhaitez retrouver une interface réseau spécifique, utilisez `Network.findNetwork()` avec un hardwareID ou un nom logique.

Retourne :

un pointeur sur un objet `YNetwork` accessible en ligne, ou `null` lorsque l'énumération est terminée.

network→ping() YNetwork

Ping l'adresse choisie pour vérifier la connexion réseau.

```
js    function ping( host)
cpp   string ping( string host)
m     -(NSString*) ping : (NSString*) host
pas   string ping( host: string): string
vb    function ping( ) As String
cs    string ping( string host)
dnp   string ping( string host)
java  String ping( String host)
uwp   async Task<string> ping( string host)
py    ping( host)
php   function ping( $host)
es    async ping( host)
cmd   YNetwork target ping host
```

Envoie quatre requêtes ICMP ECHO_REQUEST à la cible host depuis le module. Cette méthode retourne une chaîne de caractères avec le résultat des 4 requêtes ICMP ECHO_RESPONSE.

Paramètres :

host le nom d'hôte ou l'adresse IP de la cible

Retourne :

une chaîne de caractères contenant le résultat du ping.

network→registerValueCallback() YNetwork

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function **registerValueCallback**(**callback**)
cpp	int **registerValueCallback**(YNetworkValueCallback **callback**)
m	-(int) **registerValueCallback** : (YNetworkValueCallback) **callback**
pas	LongInt **registerValueCallback**(**callback**: TYNetworkValueCallback): LongInt
vb	function **registerValueCallback**( ) As Integer
cs	int **registerValueCallback**(ValueCallback **callback**)
java	int **registerValueCallback**(UpdateCallback **callback**)
uwp	async Task<int> **registerValueCallback**(ValueCallback **callback**)
py	**registerValueCallback**(**callback**)
php	function **registerValueCallback**(**$callback**)
es	async **registerValueCallback**(**callback**)

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

Paramètres :

callback la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

network→set_adminPassword() YNetwork
network→setAdminPassword()

Modifie le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module.

```
js    function set_adminPassword( newval)
cpp   int set_adminPassword( string newval)
m     -(int) setAdminPassword : (NSString*) newval
pas   integer set_adminPassword( newval: string): integer
vb    function set_adminPassword( ByVal newval As String) As Integer
cs    int set_adminPassword( string newval)
dnp   int set_adminPassword( string newval)
java  int set_adminPassword( String newval)
uwp   async Task<int> set_adminPassword( string newval)
py    set_adminPassword( newval)
php   function set_adminPassword( $newval)
es    async set_adminPassword( newval)
cmd   YNetwork target set_adminPassword newval
```

Si la valeur fournie est une chaîne vide, plus aucun mot de passe n'est nécessaire. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le mot de passe pour l'utilisateur "admin", qui devient alors instantanément nécessaire pour toute altération de l'état du module

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackCredentials() YNetwork
network→setCallbackCredentials()

Modifie le laisser-passer pour se connecter à l'adresse de callback.

js	function **set_callbackCredentials**(**newval**)
cpp	int **set_callbackCredentials**(string **newval**)
m	-(int) setCallbackCredentials : (NSString*) **newval**
pas	integer **set_callbackCredentials**(**newval**: string): integer
vb	function **set_callbackCredentials**(ByVal **newval** As String) As Integer
cs	int **set_callbackCredentials**(string **newval**)
dnp	int **set_callbackCredentials**(string **newval**)
java	int **set_callbackCredentials**(String **newval**)
uwp	async Task<int> **set_callbackCredentials**(string **newval**)
py	**set_callbackCredentials**(**newval**)
php	function **set_callbackCredentials**(**$newval**)
es	async **set_callbackCredentials**(**newval**)
cmd	YNetwork **target set_callbackCredentials newval**

Le laisser-passer doit être fourni tel que retourné par la fonction `get_callbackCredentials`, sous la forme `username:hash`. La valeur du hash dépend de la méthode d'autorisation implémentée par le callback. Pour une autorisation de type Basic, le hash est le MD5 de la chaîne `username:password`. Pour une autorisation de type Digest, le hash est le MD5 de la chaîne `username:realm:password`. Pour une utilisation simplifiée, utilisez la fonction `callbackLogin`. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le laisser-passer pour se connecter à l'adresse de callback

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackEncoding() YNetwork
network→setCallbackEncoding()

Modifie l'encodage à utiliser pour représenter les valeurs notifiées par callback.

```
js      function set_callbackEncoding( newval)
cpp     int set_callbackEncoding( Y_CALLBACKENCODING_enum newval)
m       -(int) setCallbackEncoding : (Y_CALLBACKENCODING_enum) newval
pas     integer set_callbackEncoding( newval: Integer): integer
vb      function set_callbackEncoding( ByVal newval As Integer) As Integer
cs      int set_callbackEncoding( int newval)
dnp     int set_callbackEncoding( int newval)
java    int set_callbackEncoding( int newval)
uwp     async Task<int> set_callbackEncoding( int newval)
py      set_callbackEncoding( newval)
php     function set_callbackEncoding( $newval)
es      async set_callbackEncoding( newval)
cmd     YNetwork target set_callbackEncoding newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une valeur parmi `Y_CALLBACKENCODING_FORM`, `Y_CALLBACKENCODING_JSON`, `Y_CALLBACKENCODING_JSON_ARRAY`, `Y_CALLBACKENCODING_CSV`, `Y_CALLBACKENCODING_YOCTO_API`, `Y_CALLBACKENCODING_JSON_NUM`, `Y_CALLBACKENCODING_EMONCMS`, `Y_CALLBACKENCODING_AZURE`, `Y_CALLBACKENCODING_INFLUXDB`, `Y_CALLBACKENCODING_MQTT`, `Y_CALLBACKENCODING_YOCTO_API_JZON` et `Y_CALLBACKENCODING_PRTG` représentant l'encodage à utiliser pour représenter les valeurs notifiées par callback

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackInitialDelay() YNetwork
network→setCallbackInitialDelay()

Modifie l'attente initiale avant la première notification par callback, en secondes.

```
js      function set_callbackInitialDelay( newval)
cpp     int set_callbackInitialDelay( int newval)
m       -(int) setCallbackInitialDelay : (int) newval
pas     integer set_callbackInitialDelay( newval: LongInt): integer
vb      function set_callbackInitialDelay( ByVal newval As Integer) As Integer
cs      int set_callbackInitialDelay( int newval)
dnp     int set_callbackInitialDelay( int newval)
java    int set_callbackInitialDelay( int newval)
uwp     async Task<int> set_callbackInitialDelay( int newval)
py      set_callbackInitialDelay( newval)
php     function set_callbackInitialDelay( $newval)
es      async set_callbackInitialDelay( newval)
cmd     YNetwork target set_callbackInitialDelay newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant l'attente initiale avant la première notification par callback, en secondes

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackMaxDelay() YNetwork
network→setCallbackMaxDelay()

Modifie l'attente entre deux callback HTTP lorsque rien n'est à signaler.

```
js    function set_callbackMaxDelay( newval)
cpp   int set_callbackMaxDelay( int newval)
m     -(int) setCallbackMaxDelay : (int) newval
pas   integer set_callbackMaxDelay( newval: LongInt): integer
vb    function set_callbackMaxDelay( ByVal newval As Integer) As Integer
cs    int set_callbackMaxDelay( int newval)
dnp   int set_callbackMaxDelay( int newval)
java  int set_callbackMaxDelay( int newval)
uwp   async Task<int> set_callbackMaxDelay( int newval)
py    set_callbackMaxDelay( newval)
php   function set_callbackMaxDelay( $newval)
es    async set_callbackMaxDelay( newval)
cmd   YNetwork target set_callbackMaxDelay newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant l'attente entre deux callback HTTP lorsque rien n'est à signaler

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackMethod() YNetwork
network→setCallbackMethod()

Modifie la méthode HTTP à utiliser pour signaler les changements d'état par callback.

```
js    function set_callbackMethod( newval)
cpp   int set_callbackMethod( Y_CALLBACKMETHOD_enum newval)
m     -(int) setCallbackMethod : (Y_CALLBACKMETHOD_enum) newval
pas   integer set_callbackMethod( newval: Integer): integer
vb    function set_callbackMethod( ByVal newval As Integer) As Integer
cs    int set_callbackMethod( int newval)
dnp   int set_callbackMethod( int newval)
java  int set_callbackMethod( int newval)
uwp   async Task<int> set_callbackMethod( int newval)
py    set_callbackMethod( newval)
php   function set_callbackMethod( $newval)
es    async set_callbackMethod( newval)
cmd   YNetwork target set_callbackMethod newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une valeur parmi `Y_CALLBACKMETHOD_POST`, `Y_CALLBACKMETHOD_GET` et `Y_CALLBACKMETHOD_PUT` représentant la méthode HTTP à utiliser pour signaler les changements d'état par callback

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackMinDelay() YNetwork
network→setCallbackMinDelay()

Modifie l'attente minimale entre deux callbacks HTTP, en secondes.

```
js    function set_callbackMinDelay( newval)
cpp   int set_callbackMinDelay( int newval)
m     -(int) setCallbackMinDelay : (int) newval
pas   integer set_callbackMinDelay( newval: LongInt): integer
vb    function set_callbackMinDelay( ByVal newval As Integer) As Integer
cs    int set_callbackMinDelay( int newval)
dnp   int set_callbackMinDelay( int newval)
java  int set_callbackMinDelay( int newval)
uwp   async Task<int> set_callbackMinDelay( int newval)
py    set_callbackMinDelay( newval)
php   function set_callbackMinDelay( $newval)
es    async set_callbackMinDelay( newval)
cmd   YNetwork target set_callbackMinDelay newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant l'attente minimale entre deux callbacks HTTP, en secondes

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackSchedule() YNetwork
network→setCallbackSchedule()

Modifie la planification des callbacks HTTP, sous forme de chaîne de caractères.

```
js      function set_callbackSchedule( newval)
cpp     int set_callbackSchedule( string newval)
m       -(int) setCallbackSchedule : (NSString*) newval
pas     integer set_callbackSchedule( newval: string): integer
vb      function set_callbackSchedule( ByVal newval As String) As Integer
cs      int set_callbackSchedule( string newval)
dnp     int set_callbackSchedule( string newval)
java    int set_callbackSchedule( String newval)
uwp     async Task<int> set_callbackSchedule( string newval)
py      set_callbackSchedule( newval)
php     function set_callbackSchedule( $newval)
es      async set_callbackSchedule( newval)
cmd     YNetwork target set_callbackSchedule newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant la planification des callbacks HTTP, sous forme de chaîne de caractères

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_callbackUrl() YNetwork
network→setCallbackUrl()

Modifie l'adresse (URL) de callback à notifier lors de changement d'état significatifs.

```
js      function set_callbackUrl( newval)
cpp     int set_callbackUrl( string newval)
m       -(int) setCallbackUrl : (NSString*) newval
pas     integer set_callbackUrl( newval: string): integer
vb      function set_callbackUrl( ByVal newval As String) As Integer
cs      int set_callbackUrl( string newval)
dnp     int set_callbackUrl( string newval)
java    int set_callbackUrl( String newval)
uwp     async Task<int> set_callbackUrl( string newval)
py      set_callbackUrl( newval)
php     function set_callbackUrl( $newval)
es      async set_callbackUrl( newval)
cmd     YNetwork target set_callbackUrl newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant l'adresse (URL) de callback à notifier lors de changement d'état significatifs

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_defaultPage() YNetwork
network→setDefaultPage()

Modifie la page HTML par défaut du hub.

```
js      function set_defaultPage( newval)
cpp     int set_defaultPage( string newval)
m       -(int) setDefaultPage : (NSString*) newval
pas     integer set_defaultPage( newval: string): integer
vb      function set_defaultPage( ByVal newval As String) As Integer
cs      int set_defaultPage( string newval)
dnp     int set_defaultPage( string newval)
java    int set_defaultPage( String newval)
uwp     async Task<int> set_defaultPage( string newval)
py      set_defaultPage( newval)
php     function set_defaultPage( $newval)
es      async set_defaultPage( newval)
cmd     YNetwork target set_defaultPage newval
```

Si aucune valeur n'est attribuée le hub retourne index.html qui est l'interface web du hub. Il est possible de changer cet page pour un fichier qui a été uploadé sur le hub. Attention, la taille maximale permise pour le nom de fichier est de 15 caractères. Si vous changez ce paramètre, n'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant la page HTML par défaut du hub

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_discoverable() YNetwork
network→setDiscoverable()

Modifie l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).

```
js    function set_discoverable( newval)
cpp   int set_discoverable( Y_DISCOVERABLE_enum newval)
m     -(int) setDiscoverable : (Y_DISCOVERABLE_enum) newval
pas   integer set_discoverable( newval: Integer): integer
vb    function set_discoverable( ByVal newval As Integer) As Integer
cs    int set_discoverable( int newval)
dnp   int set_discoverable( int newval)
java  int set_discoverable( int newval)
uwp   async Task<int> set_discoverable( int newval)
py    set_discoverable( newval)
php   function set_discoverable( $newval)
es    async set_discoverable( newval)
cmd   YNetwork target set_discoverable newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval soit `Y_DISCOVERABLE_FALSE`, soit `Y_DISCOVERABLE_TRUE`, selon l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour)

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_httpPort() YNetwork
network→setHttpPort()

Modifie le port TCP utilisé pour l'interface Web du hub.

```
js      function set_httpPort( newval)
cpp     int set_httpPort( int newval)
m       -(int) setHttpPort : (int) newval
pas     integer set_httpPort( newval: LongInt): integer
vb      function set_httpPort( ByVal newval As Integer) As Integer
cs      int set_httpPort( int newval)
dnp     int set_httpPort( int newval)
java    int set_httpPort( int newval)
uwp     async Task<int> set_httpPort( int newval)
py      set_httpPort( newval)
php     function set_httpPort( $newval)
es      async set_httpPort( newval)
cmd     YNetwork target set_httpPort newval
```

La valeur par défaut est le port 80, utilisé habituellement pat tous les serveurs web. Indépendamment de la valeur de ce paramètre, le hub répond toujours au port 4444, qui est utilisé par défaut par la librairie de programmation Yoctopuce. Si vous changez ce paramètre, n'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant le port TCP utilisé pour l'interface Web du hub

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_logicalName() YNetwork
network→setLogicalName()

Modifie le nom logique de l'interface réseau.

```
js      function set_logicalName( newval)
cpp     int set_logicalName( string newval)
m       -(int) setLogicalName : (NSString*) newval
pas     integer set_logicalName( newval: string): integer
vb      function set_logicalName( ByVal newval As String) As Integer
cs      int set_logicalName( string newval)
dnp     int set_logicalName( string newval)
java    int set_logicalName( String newval)
uwp     async Task<int> set_logicalName( string newval)
py      set_logicalName( newval)
php     function set_logicalName( $newval)
es      async set_logicalName( newval)
cmd     YNetwork target set_logicalName newval
```

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom logique de l'interface réseau.

Retourne :

`YAPI_SUCCESS` si l'appel se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_ntpServer() YNetwork
network→setNtpServer()

Modifie l'adresse IP du serveur NTP que le module doit utiliser.

js	function **set_ntpServer**(**newval**)
cpp	int **set_ntpServer**(string **newval**)
m	-(int) setNtpServer : (NSString*) **newval**
pas	integer **set_ntpServer**(**newval**: string): integer
vb	function **set_ntpServer**(ByVal **newval** As String) As Integer
cs	int **set_ntpServer**(string **newval**)
dnp	int **set_ntpServer**(string **newval**)
java	int **set_ntpServer**(String **newval**)
uwp	async Task<int> **set_ntpServer**(string **newval**)
py	**set_ntpServer**(**newval**)
php	function **set_ntpServer**(**$newval**)
es	async **set_ntpServer**(**newval**)
cmd	YNetwork **target set_ntpServer newval**

Utilisez une chaine vide pour restaurer l'adresse pré-configurée en usine. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

newval une chaîne de caractères représentant l'adresse IP du serveur NTP que le module doit utiliser

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_periodicCallbackSchedule() YNetwork
network→setPeriodicCallbackSchedule()

Configure la planification de callbacks HTTP périodiques (fonction simplifiée).

```
js    function set_periodicCallbackSchedule( interval, offset)
cpp   int set_periodicCallbackSchedule( string interval, int offset)
m     -(int) setPeriodicCallbackSchedule : (NSString*) interval
                                         : (int) offset
pas   LongInt set_periodicCallbackSchedule( interval: string,
                                            offset: LongInt): LongInt
vb    function set_periodicCallbackSchedule( ) As Integer
cs    int set_periodicCallbackSchedule( string interval, int offset)
dnp   int set_periodicCallbackSchedule( string interval, int offset)
java  int set_periodicCallbackSchedule( String interval, int offset)
uwp   async Task<int> set_periodicCallbackSchedule( string interval,
                                                    int offset)
py    set_periodicCallbackSchedule( interval, offset)
php   function set_periodicCallbackSchedule( $interval, $offset)
es    async set_periodicCallbackSchedule( interval, offset)
cmd   YNetwork target set_periodicCallbackSchedule interval offset
```

Paramètres :

interval la périodicité du callback sous forme textuelle, exprimée en secondes, minutes ou en heures. Par exemple: "60s", "5m", "1h", "48h".

offset un entier décrivant le décalage du callback par rapport au début de la période. Par exemple, si la périodicité est 24h, un offset de 7 déclanchera le callback chaque jour à 7h du matin.

Retourne :

YAPI_SUCCESS when the call succeeds.

On failure, throws an exception or returns a negative error code.

network→set_primaryDNS() YNetwork
network→setPrimaryDNS()

Modifie l'adresse IP du serveur de noms primaire que le module doit utiliser.

js	function **set_primaryDNS**(**newval**)
cpp	int **set_primaryDNS**(string **newval**)
m	-(int) setPrimaryDNS : (NSString*) **newval**
pas	integer **set_primaryDNS**(**newval**: string): integer
vb	function **set_primaryDNS**(ByVal **newval** As String) As Integer
cs	int **set_primaryDNS**(string **newval**)
dnp	int **set_primaryDNS**(string **newval**)
java	int **set_primaryDNS**(String **newval**)
uwp	async Task<int> **set_primaryDNS**(string **newval**)
py	**set_primaryDNS**(**newval**)
php	function **set_primaryDNS**(**$newval**)
es	async **set_primaryDNS**(**newval**)
cmd	YNetwork **target set_primaryDNS newval**

En mode DHCP, si une valeur est spécifiée, elle remplacera celle reçue du serveur DHCP. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

newval une chaîne de caractères représentant l'adresse IP du serveur de noms primaire que le module doit utiliser

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_secondaryDNS() YNetwork
network→setSecondaryDNS()

Modifie l'adresse IP du serveur de nom secondaire que le module doit utiliser.

```
js    function set_secondaryDNS( newval)
cpp   int set_secondaryDNS( string newval)
m     -(int) setSecondaryDNS : (NSString*) newval
pas   integer set_secondaryDNS( newval: string): integer
vb    function set_secondaryDNS( ByVal newval As String) As Integer
cs    int set_secondaryDNS( string newval)
dnp   int set_secondaryDNS( string newval)
java  int set_secondaryDNS( String newval)
uwp   async Task<int> set_secondaryDNS( string newval)
py    set_secondaryDNS( newval)
php   function set_secondaryDNS( $newval)
es    async set_secondaryDNS( newval)
cmd   YNetwork target set_secondaryDNS newval
```

En mode DHCP, si une valeur est spécifiée, elle remplacera celle reçue du serveur DHCP. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

newval une chaîne de caractères représentant l'adresse IP du serveur de nom secondaire que le module doit utiliser

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_userData() YNetwork
network→setUserData()

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js      function set_userData( data)
cpp     void set_userData( void * data)
m       -(void) setUserData : (id) data
pas     set_userData( data: Tobject)
vb      procedure set_userData( ByVal data As Object)
cs      void set_userData( object data)
java    void set_userData( Object data)
py      set_userData( data)
php     function set_userData( $data)
es      async set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Paramètres :

data objet quelconque à mémoriser

network→set_userPassword() YNetwork
network→setUserPassword()

Modifie le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module.

```
js    function set_userPassword( newval)
cpp   int set_userPassword( string newval)
m     -(int) setUserPassword : (NSString*) newval
pas   integer set_userPassword( newval: string): integer
vb    function set_userPassword( ByVal newval As String) As Integer
cs    int set_userPassword( string newval)
dnp   int set_userPassword( string newval)
java  int set_userPassword( String newval)
uwp   async Task<int> set_userPassword( string newval)
py    set_userPassword( newval)
php   function set_userPassword( $newval)
es    async set_userPassword( newval)
cmd   YNetwork target set_userPassword newval
```

Si la valeur fournie est une chaîne vide, plus aucun mot de passe n'est nécessaire. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le mode de passe pour l'utilisateur "user", qui devient alors instantanément nécessaire pour tout accès au module

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→set_wwwWatchdogDelay() YNetwork
network→setWwwWatchdogDelay()

Modifie la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

js	function **set_wwwWatchdogDelay**(**newval**)
cpp	int **set_wwwWatchdogDelay**(int **newval**)
m	-(int) setWwwWatchdogDelay : (int) **newval**
pas	integer **set_wwwWatchdogDelay**(**newval**: LongInt): integer
vb	function **set_wwwWatchdogDelay**(ByVal **newval** As Integer) As Integer
cs	int **set_wwwWatchdogDelay**(int **newval**)
dnp	int **set_wwwWatchdogDelay**(int **newval**)
java	int **set_wwwWatchdogDelay**(int **newval**)
uwp	async Task<int> **set_wwwWatchdogDelay**(int **newval**)
py	**set_wwwWatchdogDelay**(**newval**)
php	function **set_wwwWatchdogDelay**(**$newval**)
es	async **set_wwwWatchdogDelay**(**newval**)
cmd	YNetwork **target set_wwwWatchdogDelay newval**

Une valeur nulle désactive le redémarrage automatique en cas de perte de connectivité WWW. La plus petite durée non-nulle utilisable est 90 secondes. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant la durée de perte de connection WWW tolérée (en secondes) avant de déclancher un redémarrage automatique pour tenter de récupérer la connectivité Internet

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→triggerCallback() YNetwork

Déclenche un callback HTTP rapidement.

```
js      function triggerCallback( )
cpp     int triggerCallback( )
m       -(int) triggerCallback
pas     LongInt triggerCallback( ): LongInt
vb      function triggerCallback( ) As Integer
cs      int triggerCallback( )
dnp     int triggerCallback( )
java    int triggerCallback( )
uwp     async Task<int> triggerCallback( )
py      triggerCallback( )
php     function triggerCallback( )
es      async triggerCallback( )
cmd     YNetwork target triggerCallback
```

Cette fonction peut même être appellée à l'intérieur d'un callback HTTP, dans quel cas le callback HTTP suivant sera appelé 5 secondes après la fin du callback courant, indépendamment de l'intervalle minimal configuré dans le module.

Retourne :
une chaîne de caractères contenant le résultat du ping.

network→unmuteValueCallbacks() YNetwork

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

js	function **unmuteValueCallbacks**( )
cpp	int **unmuteValueCallbacks**( )
m	-(int) **unmuteValueCallbacks**
pas	LongInt **unmuteValueCallbacks**( ): LongInt
vb	function **unmuteValueCallbacks**( ) As Integer
cs	int **unmuteValueCallbacks**( )
dnp	int **unmuteValueCallbacks**( )
java	int **unmuteValueCallbacks**( )
uwp	async Task<int> **unmuteValueCallbacks**( )
py	**unmuteValueCallbacks**( )
php	function **unmuteValueCallbacks**( )
es	async **unmuteValueCallbacks**( )
cmd	YNetwork **target** **unmuteValueCallbacks**

Cette fonction annule un précédent appel à `muteValueCallbacks()`. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→useDHCP() YNetwork

Modifie la configuration de l'interface réseau pour utiliser une adresse assignée automatiquement par le serveur DHCP.

```
js    function useDHCP( fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)
cpp   int useDHCP( string fallbackIpAddr,
                   int fallbackSubnetMaskLen,
                   string fallbackRouter)
m     -(int) useDHCP : (NSString*) fallbackIpAddr
                     : (int) fallbackSubnetMaskLen
                     : (NSString*) fallbackRouter
pas   LongInt useDHCP( fallbackIpAddr: string,
                       fallbackSubnetMaskLen: LongInt,
                       fallbackRouter: string): LongInt
vb    function useDHCP( ) As Integer
cs    int useDHCP( string fallbackIpAddr,
                   int fallbackSubnetMaskLen,
                   string fallbackRouter)
dnp   int useDHCP( string fallbackIpAddr,
                   int fallbackSubnetMaskLen,
                   string fallbackRouter)
java  int useDHCP( String fallbackIpAddr,
                   int fallbackSubnetMaskLen,
                   String fallbackRouter)
uwp   async Task<int> useDHCP( string fallbackIpAddr,
                               int fallbackSubnetMaskLen,
                               string fallbackRouter)
py    useDHCP( fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)
php   function useDHCP( $fallbackIpAddr, $fallbackSubnetMaskLen, $fallbackRouter)
es    async useDHCP( fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)
cmd   YNetwork target useDHCP fallbackIpAddr fallbackSubnetMaskLen fallbackRouter
```

En attendant qu'une adresse soit reçue (et indéfiniment si aucun serveur DHCP ne répond), le module utilisera les paramètres IP spécifiés à cette fonction. N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

fallbackIpAddr	adresse IP à utiliser si aucun serveur DHCP ne répond
fallbackSubnetMaskLen	longueur du masque de sous-réseau à utiliser si aucun serveur DHCP ne répond. Par exemple, la valeur 24 représente 255.255.255.0.
fallbackRouter	adresse de la passerelle à utiliser si aucun serveur DHCP ne répond

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→useDHCPauto() YNetwork

Modifie la configuration de l'interface réseau pour utiliser une adresse assignée automatiquement par le serveur DHCP.

```
js    function useDHCPauto( )
cpp   int useDHCPauto( )
m     -(int) useDHCPauto
pas   LongInt useDHCPauto( ): LongInt
vb    function useDHCPauto( ) As Integer
cs    int useDHCPauto( )
dnp   int useDHCPauto( )
java  int useDHCPauto( )
uwp   async Task<int> useDHCPauto( )
py    useDHCPauto( )
php   function useDHCPauto( )
es    async useDHCPauto( )
cmd   YNetwork target useDHCPauto
```

En attendant qu'une adresse soit reçue (et indéfiniment si aucun serveur DHCP ne répond), le module utilise une adresse IP du réseau 169.254.0.0/16 (APIPA). N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→useStaticIP() YNetwork

Modifie la configuration de l'interface réseau pour utiliser une adresse IP assignée manuellement (adresse IP statique).

```
js    function useStaticIP( ipAddress, subnetMaskLen, router)
cpp   int useStaticIP( string ipAddress, int subnetMaskLen, string router)
m     -(int) useStaticIP : (NSString*) ipAddress
                         : (int) subnetMaskLen
                         : (NSString*) router
pas   LongInt useStaticIP( ipAddress: string,
                           subnetMaskLen: LongInt,
                           router: string): LongInt
vb    function useStaticIP( ) As Integer
cs    int useStaticIP( string ipAddress,
                       int subnetMaskLen,
                       string router)
dnp   int useStaticIP( string ipAddress,
                       int subnetMaskLen,
                       string router)
java  int useStaticIP( String ipAddress,
                       int subnetMaskLen,
                       String router)
uwp   async Task<int> useStaticIP( string ipAddress,
                                   int subnetMaskLen,
                                   string router)
py    useStaticIP( ipAddress, subnetMaskLen, router)
php   function useStaticIP( $ipAddress, $subnetMaskLen, $router)
es    async useStaticIP( ipAddress, subnetMaskLen, router)
cmd   YNetwork target useStaticIP ipAddress subnetMaskLen router
```

N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

ipAddress	adresse IP à utiliser par le module
subnetMaskLen	longueur du masque de sous-réseau à utiliser. Par exemple, la valeur 24 représente 255.255.255.0.
router	adresse IP de la passerelle à utiliser ("default gateway")

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

network→wait_async() YNetwork

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js  function wait_async( callback, context)
es  wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

Paramètres :

callback fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout.

8.4. La classe YFiles

Interface pour intéragir avec les systèmes de fichier, disponibles par exemple dans le Yocto-Buzzer, le Yocto-Color-V2, le YoctoHub-Ethernet et le YoctoHub-Wireless-g

La class YFiles permet d'accéder au système de fichier embarqué sur certains modules Yoctopuce. Le stockage de fichiers permet par exemple de personnaliser un service web (dans le cas d'un module connecté au réseau) ou pour d'ajouter un police de caractères (dans le cas d'un module d'affichage).

Pour utiliser les fonctions décrites ici, vous devez inclure:

js	<script type='text/javascript' src='yocto_files.js'></script>
cpp	#include "yocto_files.h"
m	#import "yocto_files.h"
pas	uses yocto_files;
vb	yocto_files.vb
cs	yocto_files.cs
dnp	import YoctoProxyAPI.YFilesProxy
java	import com.yoctopuce.YoctoAPI.YFiles;
uwp	import com.yoctopuce.YoctoAPI.YFiles;
py	from yocto_files import *
php	require_once('yocto_files.php');
es	in HTML: <script src="../../lib/yocto_files.js"></script> in node.js: require('yoctolib-es2017/yocto_files.js');
vi	YFiles.vi

Fonction globales
YFiles.FindFiles(**func**) Permet de retrouver un système de fichier d'après un identifiant donné.
YFiles.FindFilesInContext(**yctx**, **func**) Permet de retrouver un système de fichier d'après un identifiant donné dans un Context YAPI.
YFiles.FirstFiles() Commence l'énumération des systèmes de fichier accessibles par la librairie.
YFiles.FirstFilesInContext(**yctx**) Commence l'énumération des systèmes de fichier accessibles par la librairie.
YFiles.GetSimilarFunctions() Enumère toutes les fonctions de type Files disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).
Propriétés des objets `YFilesProxy`
files→AdvertisedValue *[lecture seule]* Courte chaîne de caractères représentant l'état courant de la fonction.
files→FilesCount *[lecture seule]* Nombre de fichiers présents dans le système de fichier.
files→FriendlyName *[lecture seule]* Identifiant global de la fonction au format `NOM_MODULE.NOM_FONCTION`.
files→FunctionId *[lecture seule]* Identifiant matériel du système de fichier, sans référence au module.
files→HardwareId *[lecture seule]*

Identifiant matériel unique de la fonction au format SERIAL.FUNCTIONID.

files→IsOnline *[lecture seule]*

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

files→LogicalName *[modifiable]*

Nom logique de la fonction.

files→SerialNumber *[lecture seule]*

Numéro de série du module, préprogrammé en usine.

Méthodes des objets YFiles

files→clearCache()

Invalide le cache.

files→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du système de fichier au format TYPE(NAME)=SERIAL.FUNCTIONID.

files→download(**pathname**)

Télécharge le fichier choisi du filesystème et retourne son contenu.

files→download_async(**pathname**, **callback**, **context**)

Procède au chargement du bloc suivant de mesures depuis l'enregistreur de données du module, de manière asynchrone.

files→fileExist(**filename**)

Test si un fichier esit dans le système de fichier du module.

files→format_fs()

Rétabli le système de fichier dans on état original, défragmenté.

files→get_advertisedValue()

Retourne la valeur courante du système de fichier (pas plus de 6 caractères).

files→get_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.

files→get_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.

files→get_filesCount()

Retourne le nombre de fichiers présents dans le système de fichier.

files→get_freeSpace()

Retourne l'espace disponible dans le système de fichier pour charger des nouveaux fichiers, en octets.

files→get_friendlyName()

Retourne un identifiant global du système de fichier au format NOM_MODULE.NOM_FONCTION.

files→get_functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

files→get_functionId()

Retourne l'identifiant matériel du système de fichier, sans référence au module.

files→get_hardwareId()

Retourne l'identifiant matériel unique du système de fichier au format SERIAL.FUNCTIONID.

files→get_list(**pattern**)

Retourne une liste d'objets objet YFileRecord qui décrivent les fichiers présents dans le système de fichier.

files→get_logicalName()

Retourne le nom logique du système de fichier.

files→get_module()

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

files→get_module_async(callback, context)

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

files→get_serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

files→get_userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

files→isOnline()

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

files→isOnline_async(callback, context)

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

files→isReadOnly()

Test si la fonction est en lecture seule.

files→load(msValidity)

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

files→loadAttribute(attrName)

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

files→load_async(msValidity, callback, context)

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

files→muteValueCallbacks()

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

files→nextFiles()

Continue l'énumération des systèmes de fichier commencée à l'aide de `yFirstFiles()` Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les systèmes de fichier sont retournés.

files→registerValueCallback(callback)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

files→remove(pathname)

Efface un fichier, spécifié par son path complet, du système de fichier.

files→set_logicalName(newval)

Modifie le nom logique du système de fichier.

files→set_userData(data)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

files→unmuteValueCallbacks()

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

files→upload(pathname, content)

Télécharge un contenu vers le système de fichier, au chemin d'accès spécifié.

files→wait_async(callback, context)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

YFiles.FindFiles() YFiles

YFiles.FindFiles()

Permet de retrouver un système de fichier d'après un identifiant donné.

```
js      function yFindFiles( func)
cpp     YFiles* yFindFiles( string func)
m       +(YFiles*) FindFiles : (NSString*) func
pas     TYFiles yFindFiles( func: string): TYFiles
vb      function yFindFiles( ByVal func As String) As YFiles
cs      static YFiles FindFiles( string func)
dnp     static YFilesProxy FindFiles( string func)
java    static YFiles FindFiles( String func)
uwp     static YFiles FindFiles( string func)
py      FindFiles( func)
php     function yFindFiles( $func)
es      static FindFiles( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le système de fichier soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YFiles.isOnline()` pour tester si le système de fichier est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Si un appel à la méthode is_online() de cet objet renvoie FAUX alors que vous êtes sûr que le module correspondant est bien branché, vérifiez que vous n'avez pas oublié d'appeler registerHub() à l'initialisation de de l'application.

Paramètres :

func une chaîne de caractères qui référence le système de fichier sans ambiguïté, par exemple `YBUZZER2.files`.

Retourne :

un objet de classe `YFiles` qui permet ensuite de contrôler le système de fichier.

YFiles.FindFilesInContext() YFiles
YFiles.FindFilesInContext()

Permet de retrouver un système de fichier d'après un identifiant donné dans un Context YAPI.

java static YFiles **FindFilesInContext**(YAPIContext **yctx**, String **func**)
uwp static YFiles **FindFilesInContext**(YAPIContext **yctx**, string **func**)
es static **FindFilesInContext**(**yctx**, **func**)

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le système de fichier soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YFiles.isOnline()` pour tester si le système de fichier est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Paramètres :

yctx un contexte YAPI

func une chaîne de caractères qui référence le système de fichier sans ambiguïté, par exemple `YBUZZER2.files`.

Retourne :

un objet de classe `YFiles` qui permet ensuite de contrôler le système de fichier.

YFiles.FirstFiles() YFiles

YFiles.FirstFiles()

Commence l'énumération des systèmes de fichier accessibles par la librairie.

js	function **yFirstFiles**( )
cpp	YFiles * **yFirstFiles**( )
m	+(YFiles*) **FirstFiles**
pas	TYFiles **yFirstFiles**( ): TYFiles
vb	function **yFirstFiles**( ) As YFiles
cs	static YFiles **FirstFiles**( )
java	static YFiles **FirstFiles**( )
uwp	static YFiles **FirstFiles**( )
py	**FirstFiles**( )
php	function **yFirstFiles**( )
es	static **FirstFiles**( )

Utiliser la fonction `YFiles.nextFiles()` pour itérer sur les autres systèmes de fichier.

Retourne :

un pointeur sur un objet `YFiles`, correspondant au premier système de fichier accessible en ligne, ou `null` si il n'y a pas de systèmes de fichier disponibles.

YFiles.FirstFilesInContext() YFiles
YFiles.FirstFilesInContext()

Commence l'énumération des systèmes de fichier accessibles par la librairie.

```
java  static YFiles FirstFilesInContext( YAPIContext yctx)
uwp   static YFiles FirstFilesInContext( YAPIContext yctx)
es    static FirstFilesInContext( yctx)
```

Utiliser la fonction `YFiles.nextFiles()` pour itérer sur les autres systèmes de fichier.

Paramètres :

yctx un contexte YAPI.

Retourne :

un pointeur sur un objet `YFiles`, correspondant au premier système de fichier accessible en ligne, ou `null` si il n'y a pas de systèmes de fichier disponibles.

YFiles.GetSimilarFunctions() YFiles
YFiles.GetSimilarFunctions()

Enumère toutes les fonctions de type Files disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).

dnp static new string[] **GetSimilarFunctions**( )

Chaque chaîne retournée peut être passée en argument à la méthode `YFiles.FindFiles` pour obtenir une objet permettant d'intéragir avec le module correspondant.

Retourne :

un tableau de chaînes de caractères, contenant les identifiants matériels de chaque fonction disponible trouvée.

files→AdvertisedValue YFiles

Courte chaîne de caractères représentant l'état courant de la fonction.

dnp string **AdvertisedValue**

files→FilesCount — YFiles

Nombre de fichiers présents dans le système de fichier.

dnp int **FilesCount**

files→FriendlyName YFiles

Identifiant global de la fonction au format NOM_MODULE.NOM_FONCTION.

dnp string **FriendlyName**

Le chaîne retournée utilise soit les noms logiques du module et de la fonction si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel de la fonction (par exemple: MyCustomName.relay1)

files→FunctionId — YFiles

Identifiant matériel du système de fichier, sans référence au module.

dnp string **FunctionId**

Par example `relay1`.

files→HardwareId YFiles

Identifiant matériel unique de la fonction au format SERIAL.FUNCTIONID.

dnp string **HardwareId**

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de la fonction (par example RELAYLO1-123456.relay1).

files→IsOnline YFiles

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

dnp bool **IsOnline**

Si les valeurs des attributs en cache de la fonction sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

files→LogicalName YFiles

Nom logique de la fonction.

dnp string **LogicalName**

Modifiable. Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

files→SerialNumber YFiles

Numéro de série du module, préprogrammé en usine.

dnp string **SerialNumber**

files→clearCache() YFiles

Invalide le cache.

js	function **clearCache**( )
cpp	void **clearCache**( )
m	-(void) **clearCache**
pas	**clearCache**( )
vb	procedure **clearCache**( )
cs	void **clearCache**( )
java	void **clearCache**( )
py	**clearCache**( )
php	function **clearCache**( )
es	async **clearCache**( )

Invalide le cache des valeurs courantes du système de fichier. Force le prochain appel à une méthode get_xxx() ou loadxxx() pour charger les les données depuis le module.

files→describe() YFiles

Retourne un court texte décrivant de manière non-ambigüe l'instance du système de fichier au format TYPE(NAME)=SERIAL.FUNCTIONID.

js	function **describe**( )
cpp	string **describe**( )
m	-(NSString*) **describe**
pas	string **describe**( ): string
vb	function **describe**( ) As String
cs	string **describe**( )
java	String **describe**( )
py	**describe**( )
php	function **describe**( )
es	async **describe**( )

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

Retourne :
une chaîne de caractères décrivant le système de fichier (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

files→download() YFiles

Télécharge le fichier choisi du filesystème et retourne son contenu.

```
js      function download( pathname)
cpp     string download( string pathname)
m       -(NSMutableData*) download : (NSString*) pathname
pas     TByteArray download( pathname: string): TByteArray
vb      function download( ) As Byte
cs      byte[] download( string pathname)
dnp     byte[] download( string pathname)
java    byte[] download( String pathname)
uwp     async Task<byte[]> download( string pathname)
py      download( pathname)
php     function download( $pathname)
es      async download( pathname)
cmd     YFiles target download pathname
```

Paramètres :

pathname nom complet du fichier à charger, y compris le chemin d'accès.

Retourne :

le contenu du fichier chargé sous forme d'objet binaire

En cas d'erreur, déclenche une exception ou retourne un contenu vide.

files→download_async() YFiles

Procède au chargement du bloc suivant de mesures depuis l'enregistreur de données du module, de manière asynchrone.

```
js  function download_async( pathname, callback, context)
```

Paramètres :

pathname nom complet du fichier à charger, y compris le chemin d'accès.

callback fonction fournie par l'utilisateur, qui sera appelée lorsque la suite du chargement aura été effectué. La fonction callback doit prendre trois arguments: - la variable de contexte à disposition de l'utilisateur - l'objet YFiles dont la méthode download_async a été appelée - le contenu du fichier chargé sous forme d'objet binaire

context variable de contexte à disposition de l'utilisateur

Retourne :

rien.

files→fileExist()

YFiles

Test si un fichier esit dans le système de fichier du module.

```
js      function fileExist( filename)
cpp     bool fileExist( string filename)
m       -(bool) fileExist : (NSString*) filename
pas     boolean fileExist( filename: string): boolean
vb      function fileExist( ) As Boolean
cs      bool fileExist( string filename)
dnp     bool fileExist( string filename)
java    boolean fileExist( String filename)
uwp     async Task<bool> fileExist( string filename)
py      fileExist( filename)
php     function fileExist( $filename)
es      async fileExist( filename)
cmd     YFiles target fileExist filename
```

Paramètres :

filename le nom de fichier.

Retourne :

vrais si le fichier existe, et faux is le fichier n'existe pas.

En cas d'erreur, déclenche une exception.

files→format_fs() YFiles

Rétabli le système de fichier dans on état original, défragmenté.

```
js    function format_fs( )
cpp   int format_fs( )
m     -(int) format_fs
pas   LongInt format_fs( ): LongInt
vb    function format_fs( ) As Integer
cs    int format_fs( )
dnp   int format_fs( )
java  int format_fs( )
uwp   async Task<int> format_fs( )
py    format_fs( )
php   function format_fs( )
es    async format_fs( )
cmd   YFiles target format_fs
```

entièrement vide. Tous les fichiers précédemment chargés sont irrémédiablement effacés.

Retourne :
YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

files→get_advertisedValue()
files→advertisedValue()

YFiles

Retourne la valeur courante du système de fichier (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     string get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
dnp     string get_advertisedValue( )
java    String get_advertisedValue( )
uwp     async Task<string> get_advertisedValue( )
py      get_advertisedValue( )
php     function get_advertisedValue( )
es      async get_advertisedValue( )
cmd     YFiles target get_advertisedValue
```

Retourne :
une chaîne de caractères représentant la valeur courante du système de fichier (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

files→get_errorMessage() YFiles
files→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.

```
js      function get_errorMessage( )
cpp     string get_errorMessage( )
m       -(NSString*) errorMessage
pas     string get_errorMessage( ): string
vb      function get_errorMessage( ) As String
cs      string get_errorMessage( )
java    String get_errorMessage( )
py      get_errorMessage( )
php     function get_errorMessage( )
es      get_errorMessage( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du système de fichier.

files→get_errorType() YFiles
files→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du système de fichier.

```
js      function get_errorType( )
cpp     YRETCODE get_errorType( )
m       -(YRETCODE) errorType
pas     YRETCODE get_errorType( ): YRETCODE
vb      function get_errorType( ) As YRETCODE
cs      YRETCODE get_errorType( )
java    int get_errorType( )
py      get_errorType( )
php     function get_errorType( )
es      get_errorType( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du système de fichier.

files→get_filesCount()
files→filesCount()

YFiles

Retourne le nombre de fichiers présents dans le système de fichier.

```
js      function get_filesCount( )
cpp     int get_filesCount( )
m       -(int) filesCount
pas     LongInt get_filesCount( ): LongInt
vb      function get_filesCount( ) As Integer
cs      int get_filesCount( )
dnp     int get_filesCount( )
java    int get_filesCount( )
uwp     async Task<int> get_filesCount( )
py      get_filesCount( )
php     function get_filesCount( )
es      async get_filesCount( )
cmd     YFiles target get_filesCount
```

Retourne :
un entier représentant le nombre de fichiers présents dans le système de fichier

En cas d'erreur, déclenche une exception ou retourne `Y_FILESCOUNT_INVALID`.

files→get_freeSpace()
files→freeSpace()

YFiles

Retourne l'espace disponible dans le système de fichier pour charger des nouveaux fichiers, en octets.

```
js      function get_freeSpace( )
cpp     int get_freeSpace( )
m       -(int) freeSpace
pas     LongInt get_freeSpace( ): LongInt
vb      function get_freeSpace( ) As Integer
cs      int get_freeSpace( )
dnp     int get_freeSpace( )
java    int get_freeSpace( )
uwp     async Task<int> get_freeSpace( )
py      get_freeSpace( )
php     function get_freeSpace( )
es      async get_freeSpace( )
cmd     YFiles target get_freeSpace
```

Retourne :

un entier représentant l'espace disponible dans le système de fichier pour charger des nouveaux fichiers, en octets

En cas d'erreur, déclenche une exception ou retourne `Y_FREESPACE_INVALID`.

files→get_friendlyName() YFiles
files→friendlyName()

Retourne un identifiant global du système de fichier au format NOM_MODULE.NOM_FONCTION.

js	function **get_friendlyName**( )
cpp	string **get_friendlyName**( )
m	-(NSString*) friendlyName
cs	string **get_friendlyName**( )
dnp	string **get_friendlyName**( )
java	String **get_friendlyName**( )
py	**get_friendlyName**( )
php	function **get_friendlyName**( )
es	async **get_friendlyName**( )

Le chaîne retournée utilise soit les noms logiques du module et du système de fichier si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel du système de fichier (par exemple: MyCustomName.relay1)

Retourne :

une chaîne de caractères identifiant le système de fichier en utilisant les noms logiques (ex: MyCustomName.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

files→get_functionDescriptor() YFiles
files→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

```
js      function get_functionDescriptor( )
cpp     YFUN_DESCR get_functionDescriptor( )
m       -(YFUN_DESCR) functionDescriptor
pas     YFUN_DESCR get_functionDescriptor( ): YFUN_DESCR
vb      function get_functionDescriptor( ) As YFUN_DESCR
cs      YFUN_DESCR get_functionDescriptor( )
java    String get_functionDescriptor( )
py      get_functionDescriptor( )
php     function get_functionDescriptor( )
es      async get_functionDescriptor( )
```

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

Retourne :
un identifiant de type YFUN_DESCR.

Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

files→get_functionId() YFiles
files→functionId()

Retourne l'identifiant matériel du système de fichier, sans référence au module.

```
js      function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
dnp     string get_functionId( )
java    String get_functionId( )
py      get_functionId( )
php     function get_functionId( )
es      async get_functionId( )
```

Par example `relay1`.

Retourne :
une chaîne de caractères identifiant le système de fichier (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

files→get_hardwareId()
files→hardwareId()

YFiles

Retourne l'identifiant matériel unique du système de fichier au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
dnp     string get_hardwareId( )
java    String get_hardwareId( )
py      get_hardwareId( )
php     function get_hardwareId( )
es      async get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du système de fichier (par example RELAYLO1-123456.relay1).

Retourne :
une chaîne de caractères identifiant le système de fichier (ex: RELAYLO1-123456.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

files→get_list() YFiles
files→list()

Retourne une liste d'objets objet YFileRecord qui décrivent les fichiers présents dans le système de fichier.

js	function **get_list**(**pattern**)
cpp	vector<YFileRecord> **get_list**(string **pattern**)
m	-(NSMutableArray*) list : (NSString*) **pattern**
pas	TYFileRecordArray **get_list**(**pattern**: string): TYFileRecordArray
vb	function **get_list**( ) As List
cs	List<YFileRecord> **get_list**(string **pattern**)
dnp	YFileRecordProxy[] **get_list**(string **pattern**)
java	ArrayList<YFileRecord> **get_list**(String **pattern**)
uwp	async Task<List<YFileRecord>> **get_list**(string **pattern**)
py	**get_list**(**pattern**)
php	function **get_list**(**$pattern**)
es	async **get_list**(**pattern**)
cmd	YFiles **target get_list pattern**

Paramètres :

pattern un filtre optionel sur les noms de fichiers retournés, pouvant contenir des astérisques et des points d'interrogations comme jokers. Si le pattern fourni est vide, tous les fichiers sont retournés.

Retourne :

une liste d'objets `YFileRecord`, contenant le nom complet (y compris le chemin d'accès), la taille en octets et le CRC 32-bit du contenu du fichier.

En cas d'erreur, déclenche une exception ou retourne une liste vide.

files→get_logicalName()
files→logicalName()

YFiles

Retourne le nom logique du système de fichier.

```
js      function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     string get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
dnp     string get_logicalName( )
java    String get_logicalName( )
uwp     async Task<string> get_logicalName( )
py      get_logicalName( )
php     function get_logicalName( )
es      async get_logicalName( )
cmd     YFiles target get_logicalName
```

Retourne :
une chaîne de caractères représentant le nom logique du système de fichier.

En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

files→get_module()
files→module()

YFiles

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js      function get_module( )
cpp     YModule * get_module( )
m       -(YModule*) module
pas     TYModule get_module( ): TYModule
vb      function get_module( ) As YModule
cs      YModule get_module( )
dnp     YModuleProxy get_module( )
java    YModule get_module( )
py      get_module( )
php     function get_module( )
es      async get_module( )
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Retourne :
une instance de YModule

files→get_module_async() YFiles
files→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

files→get_serialNumber() YFiles
files→serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

```
js      function get_serialNumber( )
cpp     string get_serialNumber( )
m       -(NSString*) serialNumber
pas     string get_serialNumber( ): string
vb      function get_serialNumber( ) As String
cs      string get_serialNumber( )
dnp     string get_serialNumber( )
java    String get_serialNumber( )
uwp     async Task<string> get_serialNumber( )
py      get_serialNumber( )
php     function get_serialNumber( )
es      async get_serialNumber( )
cmd     YFiles target get_serialNumber
```

Retourne :

: une chaîne de caractères représentant le numéro de série du module, préprogrammé en usine.

En cas d'erreur, déclenche une exception ou retourne YModule.SERIALNUMBER_INVALID.

files→get_userData() YFiles
files→userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

```
js      function get_userData( )
cpp     void * get_userData( )
m       -(id) userData
pas     Tobject get_userData( ): Tobject
vb      function get_userData( ) As Object
cs      object get_userData( )
java    Object get_userData( )
py      get_userData( )
php     function get_userData( )
es      async get_userData( )
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Retourne :
l'objet stocké précédemment par l'appelant.

files→isOnline() YFiles

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     boolean isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
dnp     bool isOnline( )
java    boolean isOnline( )
py      isOnline( )
php     function isOnline( )
es      async isOnline( )
```

Si les valeurs des attributs en cache du système de fichier sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Retourne :
`true` si le système de fichier est joignable, `false` sinon

files→isOnline_async() YFiles

Vérifie si le module hébergeant le système de fichier est joignable, sans déclencher d'erreur.

```
js  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du système de fichier sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

files→isReadOnly() YFiles

Test si la fonction est en lecture seule.

cpp	bool **isReadOnly**( )
m	-(bool) **isReadOnly**
pas	boolean **isReadOnly**( ): boolean
vb	function **isReadOnly**( ) As Boolean
cs	bool **isReadOnly**( )
dnp	bool **isReadOnly**( )
java	boolean **isReadOnly**( )
uwp	async Task<bool> **isReadOnly**( )
py	**isReadOnly**( )
php	function **isReadOnly**( )
es	async **isReadOnly**( )
cmd	YFiles **target isReadOnly**

Retourne vrais si la fonction est protégé en ecriture ou que la fontion n'est pas disponible.

Retourne :
`true` si la fonction est protégé en ecriture ou que la fontion n'est pas disponible

files→load() YFiles

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

```
js      function load( msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (u64) msValidity
pas     YRETCODE load( msValidity: u64): YRETCODE
vb      function load( ByVal msValidity As Long) As YRETCODE
cs      YRETCODE load( ulong msValidity)
java    int load( long msValidity)
py      load( msValidity)
php     function load( $msValidity)
es      async load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

files→loadAttribute() YFiles

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

```
js      function loadAttribute( attrName)
cpp     string loadAttribute( string attrName)
m       -(NSString*) loadAttribute : (NSString*) attrName
pas     string loadAttribute( attrName: string): string
vb      function loadAttribute( ) As String
cs      string loadAttribute( string attrName)
dnp     string loadAttribute( string attrName)
java    String loadAttribute( String attrName)
uwp     async Task<string> loadAttribute( string attrName)
py      loadAttribute( attrName)
php     function loadAttribute( $attrName)
es      async loadAttribute( attrName)
```

Paramètres :

attrName le nom de l'attribut désiré

Retourne :

une chaîne de caractères représentant la valeur actuelle de l'attribut.

En cas d'erreur, déclenche une exception ou retourne un chaîne vide.

files→load_async() YFiles

Met en cache les valeurs courantes du système de fichier, avec une durée de validité spécifiée.

```
js  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

msValidity	un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
callback	fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)
context	contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

files→muteValueCallbacks() YFiles

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

js	function **muteValueCallbacks**( )
cpp	int **muteValueCallbacks**( )
m	-(int) **muteValueCallbacks**
pas	LongInt **muteValueCallbacks**( ): LongInt
vb	function **muteValueCallbacks**( ) As Integer
cs	int **muteValueCallbacks**( )
dnp	int **muteValueCallbacks**( )
java	int **muteValueCallbacks**( )
uwp	async Task<int> **muteValueCallbacks**( )
py	**muteValueCallbacks**( )
php	function **muteValueCallbacks**( )
es	async **muteValueCallbacks**( )
cmd	YFiles **target muteValueCallbacks**

Vous pouvez utiliser cette fonction pour économiser la bande passante et le CPU sur les machines de faible puissance, ou pour éviter le déclanchement de callbacks HTTP. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

files→nextFiles() YFiles

Continue l'énumération des systèmes de fichier commencée à l'aide de `yFirstFiles()`. Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les systèmes de fichier sont retournés.

```
js    function nextFiles( )
cpp   YFiles * nextFiles( )
m     -(YFiles*) nextFiles
pas   TYFiles nextFiles( ): TYFiles
vb    function nextFiles( ) As YFiles
cs    YFiles nextFiles( )
java  YFiles nextFiles( )
uwp   YFiles nextFiles( )
py    nextFiles( )
php   function nextFiles( )
es    nextFiles( )
```

Si vous souhaitez retrouver un système de fichier spécifique, utilisez `Files.findFiles()` avec un hardwareID ou un nom logique.

Retourne :
un pointeur sur un objet `YFiles` accessible en ligne, ou `null` lorsque l'énumération est terminée.

files→registerValueCallback() YFiles

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

```
js      function registerValueCallback( callback)
cpp     int registerValueCallback( YFilesValueCallback callback)
m       -(int) registerValueCallback : (YFilesValueCallback) callback
pas     LongInt registerValueCallback( callback: TYFilesValueCallback): LongInt
vb      function registerValueCallback( ) As Integer
cs      int registerValueCallback( ValueCallback callback)
java    int registerValueCallback( UpdateCallback callback)
uwp     async Task<int> registerValueCallback( ValueCallback callback)
py      registerValueCallback( callback)
php     function registerValueCallback( $callback)
es      async registerValueCallback( callback)
```

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

Paramètres :

callback la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

files→remove() YFiles

Efface un fichier, spécifié par son path complet, du système de fichier.

```
js      function remove( pathname)
cpp     int remove( string pathname)
m       -(int) remove : (NSString*) pathname
pas     LongInt remove( pathname: string): LongInt
vb      function remove( ) As Integer
cs      int remove( string pathname)
dnp     int remove( string pathname)
java    int remove( String pathname)
uwp     async Task<int> remove( string pathname)
py      remove( pathname)
php     function remove( $pathname)
es      async remove( pathname)
cmd     YFiles target remove pathname
```

A cause de la fragmentation, l'effacement d'un fichier ne libère pas toujours la totalité de l'espace qu'il occuppe. Par contre, la ré-écriture d'un fichier du même nom récupérera dans tout les cas l'espace qui n'aurait éventuellement pas été libéré. Pour s'assurer de libérer la totalité de l'espace du système de fichier, utilisez la fonction `format_fs`.

Paramètres :

pathname nom complet du fichier, y compris le chemin d'accès.

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

files→set_logicalName() YFiles
files→setLogicalName()

Modifie le nom logique du système de fichier.

js	function **set_logicalName**(**newval**)
cpp	int **set_logicalName**(string **newval**)
m	-(int) setLogicalName : (NSString*) **newval**
pas	integer **set_logicalName**(**newval**: string): integer
vb	function **set_logicalName**(ByVal **newval** As String) As Integer
cs	int **set_logicalName**(string **newval**)
dnp	int **set_logicalName**(string **newval**)
java	int **set_logicalName**(String **newval**)
uwp	async Task<int> **set_logicalName**(string **newval**)
py	**set_logicalName**(**newval**)
php	function **set_logicalName**(**$newval**)
es	async **set_logicalName**(**newval**)
cmd	YFiles **target set_logicalName newval**

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom logique du système de fichier.

Retourne :

`YAPI_SUCCESS` si l'appel se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

files→set_userData() YFiles
files→setUserData()

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js    function set_userData( data)
cpp   void set_userData( void * data)
m     -(void) setUserData : (id) data
pas   set_userData( data: Tobject)
vb    procedure set_userData( ByVal data As Object)
cs    void set_userData( object data)
java  void set_userData( Object data)
py    set_userData( data)
php   function set_userData( $data)
es    async set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Paramètres :

data objet quelconque à mémoriser

files→unmuteValueCallbacks() YFiles

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

```
js      function unmuteValueCallbacks( )
cpp     int unmuteValueCallbacks( )
m       -(int) unmuteValueCallbacks
pas     LongInt unmuteValueCallbacks( ): LongInt
vb      function unmuteValueCallbacks( ) As Integer
cs      int unmuteValueCallbacks( )
dnp     int unmuteValueCallbacks( )
java    int unmuteValueCallbacks( )
uwp     async Task<int> unmuteValueCallbacks( )
py      unmuteValueCallbacks( )
php     function unmuteValueCallbacks( )
es      async unmuteValueCallbacks( )
cmd     YFiles target unmuteValueCallbacks
```

Cette fonction annule un précédent appel à `muteValueCallbacks()`. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

files→upload() YFiles

Télécharge un contenu vers le système de fichier, au chemin d'accès spécifié.

```
js    function upload( pathname, content)
cpp   int upload( string pathname, string content)
m     -(int) upload : (NSString*) pathname
                    : (NSData*) content
pas   LongInt upload( pathname: string, content: TByteArray): LongInt
vb    procedure upload( )
cs    int upload( string pathname)
dnp   int upload( string pathname)
java  int upload( String pathname, byte[] content)
uwp   async Task<int> upload( string pathname)
py    upload( pathname, content)
php   function upload( $pathname, $content)
es    async upload( pathname, content)
cmd   YFiles target upload pathname content
```

Si un fichier existe déjà pour le même chemin d'accès, son contenu est remplacé.

Paramètres :

pathname nom complet du fichier, y compris le chemin d'accès.

content contenu du fichier à télécharger

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

files→wait_async() YFiles

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js  function wait_async( callback, context)
es  wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

Paramètres :

callback fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout.

8.5. La classe YRealTimeClock

Interface pour intéragir avec les horloges à temps réel, disponibles par exemple dans le YoctoHub-GSM-3G-EU, le YoctoHub-GSM-3G-NA, le YoctoHub-Wireless-SR et le YoctoHub-Wireless-g

La classe `YRealTimeClock` permet d'accéder à l'horloge embarquée sur certains modules Yoctopuce. Elle fournit la date et l'heure courante de manière persistante, même en cas de coupure de courant de plusieurs jours. Elle est le fondement des fonctions de réveil automatique implémentées par le WakeUpScheduler. L'heure courante peut représenter aussi bien une heure locale qu'une heure UTC, mais aucune adaptation automatique n'est fait au changement d'heure été/hiver.

Pour utiliser les fonctions décrites ici, vous devez inclure:

es	in HTML: <script src="../../lib/yocto_realtimeclock.js"></script> in node.js: require('yoctolib-es2017/yocto_realtimeclock.js');
js	<script type='text/javascript' src='yocto_realtimeclock.js'></script>
cpp	#include "yocto_realtimeclock.h"
m	#import "yocto_realtimeclock.h"
pas	uses yocto_realtimeclock;
vb	yocto_realtimeclock.vb
cs	yocto_realtimeclock.cs
dnp	import YoctoProxyAPI.YRealTimeClockProxy
java	import com.yoctopuce.YoctoAPI.YRealTimeClock;
uwp	import com.yoctopuce.YoctoAPI.YRealTimeClock;
py	from yocto_realtimeclock import *
php	require_once('yocto_realtimeclock.php');
vi	YRealTimeClock.vi

Fonction globales

YRealTimeClock.FindRealTimeClock(**func**)

Permet de retrouver une horloge à temps réel d'après un identifiant donné.

YRealTimeClock.FindRealTimeClockInContext(**yctx**, **func**)

Permet de retrouver une horloge à temps réel d'après un identifiant donné dans un Context YAPI.

YRealTimeClock.FirstRealTimeClock()

Commence l'énumération des horloges à temps réel accessibles par la librairie.

YRealTimeClock.FirstRealTimeClockInContext(**yctx**)

Commence l'énumération des horloges à temps réel accessibles par la librairie.

YRealTimeClock.GetSimilarFunctions()

Enumère toutes les fonctions de type RealTimeClock disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).

Propriétés des objets `YRealTimeClockProxy`

realtimeclock→AdvertisedValue *[lecture seule]*

Courte chaîne de caractères représentant l'état courant de la fonction.

realtimeclock→FriendlyName *[lecture seule]*

Identifiant global de la fonction au format `NOM_MODULE.NOM_FONCTION`.

realtimeclock→FunctionId *[lecture seule]*

Identifiant matériel de l'horloge à temps réel, sans référence au module.

realtimeclock→HardwareId *[lecture seule]*

Identifiant matériel unique de la fonction au format `SERIAL.FUNCTIONID`.

realtimeclock→IsOnline *[lecture seule]*

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

realtimeclock→LogicalName *[modifiable]*

Nom logique de la fonction.

realtimeclock→SerialNumber *[lecture seule]*

Numéro de série du module, préprogrammé en usine.

realtimeclock→UtcOffset *[modifiable]*

Nombre de secondes de décallage entre l'heure courante et l'heure UTC (time zone).

Méthodes des objets `YRealTimeClock`

realtimeclock→clearCache()

Invalide le cache.

realtimeclock→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'horloge à temps réel au format `TYPE(NAME)=SERIAL.FUNCTIONID`.

realtimeclock→get_advertisedValue()

Retourne la valeur courante de l'horloge à temps réel (pas plus de 6 caractères).

realtimeclock→get_dateTime()

Retourne l'heure courante au format "AAAA/MM/JJ hh:mm:ss".

realtimeclock→get_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'horloge à temps réel.

realtimeclock→get_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'horloge à temps réel.

realtimeclock→get_friendlyName()

Retourne un identifiant global de l'horloge à temps réel au format `NOM_MODULE.NOM_FONCTION`.

realtimeclock→get_functionDescriptor()

Retourne un identifiant unique de type `YFUN_DESCR` correspondant à la fonction.

realtimeclock→get_functionId()

Retourne l'identifiant matériel de l'horloge à temps réel, sans référence au module.

realtimeclock→get_hardwareId()

Retourne l'identifiant matériel unique de l'horloge à temps réel au format `SERIAL.FUNCTIONID`.

realtimeclock→get_logicalName()

Retourne le nom logique de l'horloge à temps réel.

realtimeclock→get_module()

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

realtimeclock→get_module_async(**callback**, **context**)

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

realtimeclock→get_serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

realtimeclock→get_timeSet()

Retourne vrai si l'horloge à été mise à l'heure, sinon faux.

realtimeclock→get_unixTime()

Retourne l'heure courante au format Unix (nombre de seconds secondes écoulées depuis le 1er janvier 1970).

realtimeclock→get_userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

realtimeclock→get_utcOffset()

Retourne le nombre de secondes de décallage entre l'heure courante et l'heure UTC (time zone).

realtimeclock→isOnline()

Vérifie si le module hébergeant l'horloge à temps réel est joignable, sans déclencher d'erreur.

realtimeclock→isOnline_async(**callback**, **context**)

Vérifie si le module hébergeant l'horloge à temps réel est joignable, sans déclencher d'erreur.

realtimeclock→isReadOnly()

Test si la fonction est en lecture seule.

realtimeclock→load(**msValidity**)

Met en cache les valeurs courantes de l'horloge à temps réel, avec une durée de validité spécifiée.

realtimeclock→loadAttribute(**attrName**)

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

realtimeclock→load_async(**msValidity**, **callback**, **context**)

Met en cache les valeurs courantes de l'horloge à temps réel, avec une durée de validité spécifiée.

realtimeclock→muteValueCallbacks()

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

realtimeclock→nextRealTimeClock()

Continue l'énumération des horloges à temps réel commencée à l'aide de `yFirstRealTimeClock()`
Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les horloges à temps réel sont retournés.

realtimeclock→registerValueCallback(**callback**)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

realtimeclock→set_logicalName(**newval**)

Modifie le nom logique de l'horloge à temps réel.

realtimeclock→set_unixTime(**newval**)

Modifie l'heure courante.

realtimeclock→set_userData(**data**)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

realtimeclock→set_utcOffset(**newval**)

Modifie le nombre de secondes de décalage entre l'heure courante et l'heure UTC (time zone).

realtimeclock→unmuteValueCallbacks()

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

realtimeclock→wait_async(**callback**, **context**)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

YRealTimeClock.FindRealTimeClock() YRealTimeClock
YRealTimeClock.FindRealTimeClock()

Permet de retrouver une horloge à temps réel d'après un identifiant donné.

```
js    function yFindRealTimeClock( func)
cpp   YRealTimeClock* yFindRealTimeClock( string func)
m     +(YRealTimeClock*) FindRealTimeClock : (NSString*) func
pas   TYRealTimeClock yFindRealTimeClock( func: string): TYRealTimeClock
vb    function yFindRealTimeClock( ByVal func As String) As YRealTimeClock
cs    static YRealTimeClock FindRealTimeClock( string func)
dnp   static YRealTimeClockProxy FindRealTimeClock( string func)
java  static YRealTimeClock FindRealTimeClock( String func)
uwp   static YRealTimeClock FindRealTimeClock( string func)
py    FindRealTimeClock( func)
php   function yFindRealTimeClock( $func)
es    static FindRealTimeClock( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'horloge à temps réel soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YRealTimeClock.isOnline()` pour tester si l'horloge à temps réel est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Si un appel à la méthode is_online() de cet objet renvoie FAUX alors que vous êtes sûr que le module correspondant est bien branché, vérifiez que vous n'avez pas oublié d'appeler registerHub() à l'initialisation de de l'application.

Paramètres :

func une chaîne de caractères qui référence l'horloge à temps réel sans ambiguïté, par exemple `YHUBGSM3.realTimeClock`.

Retourne :

un objet de classe `YRealTimeClock` qui permet ensuite de contrôler l'horloge à temps réel.

YRealTimeClock.FindRealTimeClockInContext() YRealTimeClock
YRealTimeClock.FindRealTimeClockInContext()

Permet de retrouver une horloge à temps réel d'après un identifiant donné dans un Context YAPI.

```
java  static YRealTimeClock FindRealTimeClockInContext( YAPIContext yctx,
                                                        String func)
uwp   static YRealTimeClock FindRealTimeClockInContext( YAPIContext yctx,
                                                        string func)
es    static FindRealTimeClockInContext( yctx, func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que l'horloge à temps réel soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YRealTimeClock.isOnline()` pour tester si l'horloge à temps réel est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Paramètres :

yctx un contexte YAPI

func une chaîne de caractères qui référence l'horloge à temps réel sans ambiguïté, par exemple `YHUBGSM3.realTimeClock`.

Retourne :

un objet de classe `YRealTimeClock` qui permet ensuite de contrôler l'horloge à temps réel.

YRealTimeClock.FirstRealTimeClock() YRealTimeClock
YRealTimeClock.FirstRealTimeClock()

Commence l'énumération des horloges à temps réel accessibles par la librairie.

js	function **yFirstRealTimeClock**( )
cpp	YRealTimeClock * **yFirstRealTimeClock**( )
m	+(YRealTimeClock*) **FirstRealTimeClock**
pas	TYRealTimeClock **yFirstRealTimeClock**( ): TYRealTimeClock
vb	function **yFirstRealTimeClock**( ) As YRealTimeClock
cs	static YRealTimeClock **FirstRealTimeClock**( )
java	static YRealTimeClock **FirstRealTimeClock**( )
uwp	static YRealTimeClock **FirstRealTimeClock**( )
py	**FirstRealTimeClock**( )
php	function **yFirstRealTimeClock**( )
es	static **FirstRealTimeClock**( )

Utiliser la fonction `YRealTimeClock.nextRealTimeClock()` pour itérer sur les autres horloges à temps réel.

Retourne :

un pointeur sur un objet `YRealTimeClock`, correspondant à la première horloge à temps réel accessible en ligne, ou `null` si il n'y a pas de horloges à temps réel disponibles.

YRealTimeClock.FirstRealTimeClockInContext() YRealTimeClock
YRealTimeClock.FirstRealTimeClockInContext()

Commence l'énumération des horloges à temps réel accessibles par la librairie.

java	static YRealTimeClock **FirstRealTimeClockInContext**(YAPIContext **yctx**)
uwp	static YRealTimeClock **FirstRealTimeClockInContext**(YAPIContext **yctx**)
es	static **FirstRealTimeClockInContext**(**yctx**)

Utiliser la fonction `YRealTimeClock.nextRealTimeClock()` pour itérer sur les autres horloges à temps réel.

Paramètres :

yctx un contexte YAPI.

Retourne :

un pointeur sur un objet `YRealTimeClock`, correspondant à la première horloge à temps réel accessible en ligne, ou `null` si il n'y a pas de horloges à temps réel disponibles.

YRealTimeClock.GetSimilarFunctions() YRealTimeClock
YRealTimeClock.GetSimilarFunctions()

Enumère toutes les fonctions de type RealTimeClock disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).

dnp static new string[] **GetSimilarFunctions**( )

Chaque chaîne retournée peut être passée en argument à la méthode `YRealTimeClock.FindRealTimeClock` pour obtenir une objet permettant d'intéragir avec le module correspondant.

Retourne :
un tableau de chaînes de caractères, contenant les identifiants matériels de chaque fonction disponible trouvée.

realtimeclock→AdvertisedValue YRealTimeClock

Courte chaîne de caractères représentant l'état courant de la fonction.

dnp string **AdvertisedValue**

realtimeclock→FriendlyName — YRealTimeClock

Identifiant global de la fonction au format NOM_MODULE.NOM_FONCTION.

dnp string **FriendlyName**

Le chaîne retournée utilise soit les noms logiques du module et de la fonction si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel de la fonction (par exemple: MyCustomName.relay1)

realtimeclock→FunctionId YRealTimeClock

Identifiant matériel de l'horloge à temps réel, sans référence au module.

dnp string **FunctionId**

Par example `relay1`.

realtimeclock→HardwareId — YRealTimeClock

Identifiant matériel unique de la fonction au format SERIAL.FUNCTIONID.

dnp string **HardwareId**

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de la fonction (par example RELAYLO1-123456.relay1).

realtimeclock→IsOnline — YRealTimeClock

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

dnp bool **IsOnline**

Si les valeurs des attributs en cache de la fonction sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

realtimeclock→LogicalName YRealTimeClock

Nom logique de la fonction.

dnp string **LogicalName**

Modifiable. Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

realtimeclock→SerialNumber — YRealTimeClock

Numéro de série du module, préprogrammé en usine.

dnp string **SerialNumber**

realtimeclock→UtcOffset — YRealTimeClock

Nombre de secondes de décallage entre l'heure courante et l'heure UTC (time zone).

dnp int **UtcOffset**

Modifiable. Modifie le nombre de secondes de décalage entre l'heure courante et l'heure UTC (time zone). Le décallage est automatiquement arrondi au quart d'heure le plus proche. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

realtimeclock→clearCache() YRealTimeClock

Invalide le cache.

js	function **clearCache**( )
cpp	void **clearCache**( )
m	-(void) **clearCache**
pas	**clearCache**( )
vb	procedure **clearCache**( )
cs	void **clearCache**( )
java	void **clearCache**( )
py	**clearCache**( )
php	function **clearCache**( )
es	async **clearCache**( )

Invalide le cache des valeurs courantes de l'horloge à temps réel. Force le prochain appel à une méthode get_xxx() ou loadxxx() pour charger les les données depuis le module.

realtimeclock→describe() YRealTimeClock

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'horloge à temps réel au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     string describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      describe( )
php     function describe( )
es      async describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

Retourne :
une chaîne de caractères décrivant l'horloge à temps réel (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

realtimeclock→get_advertisedValue() YRealTimeClock
realtimeclock→advertisedValue()

Retourne la valeur courante de l'horloge à temps réel (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     string get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
dnp     string get_advertisedValue( )
java    String get_advertisedValue( )
uwp     async Task<string> get_advertisedValue( )
py      get_advertisedValue( )
php     function get_advertisedValue( )
es      async get_advertisedValue( )
cmd     YRealTimeClock target get_advertisedValue
```

Retourne :

une chaîne de caractères représentant la valeur courante de l'horloge à temps réel (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

realtimeclock→get_dateTime()
realtimeclock→dateTime()

YRealTimeClock

Retourne l'heure courante au format "AAAA/MM/JJ hh:mm:ss".

```
js    function get_dateTime( )
cpp   string get_dateTime( )
m     -(NSString*) dateTime
pas   string get_dateTime( ): string
vb    function get_dateTime( ) As String
cs    string get_dateTime( )
dnp   string get_dateTime( )
java  String get_dateTime( )
uwp   async Task<string> get_dateTime( )
py    get_dateTime( )
php   function get_dateTime( )
es    async get_dateTime( )
cmd   YRealTimeClock target get_dateTime
```

Retourne :

une chaîne de caractères représentant l'heure courante au format "AAAA/MM/JJ hh:mm:ss"

En cas d'erreur, déclenche une exception ou retourne `Y_DATETIME_INVALID`.

realtimeclock→get_errorMessage() YRealTimeClock
realtimeclock→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation de l'horloge à temps réel.

```
js      function get_errorMessage( )
cpp     string get_errorMessage( )
m       -(NSString*) errorMessage
pas     string get_errorMessage( ): string
vb      function get_errorMessage( ) As String
cs      string get_errorMessage( )
java    String get_errorMessage( )
py      get_errorMessage( )
php     function get_errorMessage( )
es      get_errorMessage( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation de l'horloge à temps réel.

realtimeclock→get_errorType() YRealTimeClock
realtimeclock→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation de l'horloge à temps réel.

js	function **get_errorType**( )
cpp	YRETCODE **get_errorType**( )
m	-(YRETCODE) errorType
pas	YRETCODE **get_errorType**( ): YRETCODE
vb	function **get_errorType**( ) As YRETCODE
cs	YRETCODE **get_errorType**( )
java	int **get_errorType**( )
py	**get_errorType**( )
php	function **get_errorType**( )
es	**get_errorType**( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation de l'horloge à temps réel.

realtimeclock→get_friendlyName() YRealTimeClock
realtimeclock→friendlyName()

Retourne un identifiant global de l'horloge à temps réel au format NOM_MODULE.NOM_FONCTION.

```
js      function get_friendlyName( )
cpp     string get_friendlyName( )
m       -(NSString*) friendlyName
cs      string get_friendlyName( )
dnp     string get_friendlyName( )
java    String get_friendlyName( )
py      get_friendlyName( )
php     function get_friendlyName( )
es      async get_friendlyName( )
```

Le chaîne retournée utilise soit les noms logiques du module et de l'horloge à temps réel si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel de l'horloge à temps réel (par exemple: MyCustomName.relay1)

Retourne :

une chaîne de caractères identifiant l'horloge à temps réel en utilisant les noms logiques (ex: MyCustomName.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

realtimeclock→get_functionDescriptor() YRealTimeClock
realtimeclock→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function **get_functionDescriptor**( )
cpp	YFUN_DESCR **get_functionDescriptor**( )
m	-(YFUN_DESCR) functionDescriptor
pas	YFUN_DESCR **get_functionDescriptor**( ): YFUN_DESCR
vb	function **get_functionDescriptor**( ) As YFUN_DESCR
cs	YFUN_DESCR **get_functionDescriptor**( )
java	String **get_functionDescriptor**( )
py	**get_functionDescriptor**( )
php	function **get_functionDescriptor**( )
es	async **get_functionDescriptor**( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

Retourne :
un identifiant de type YFUN_DESCR.

Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

realtimeclock→get_functionId() YRealTimeClock
realtimeclock→functionId()

Retourne l'identifiant matériel de l'horloge à temps réel, sans référence au module.

```
js      function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
dnp     string get_functionId( )
java    String get_functionId( )
py      get_functionId( )
php     function get_functionId( )
es      async get_functionId( )
```

Par example `relay1`.

Retourne :
une chaîne de caractères identifiant l'horloge à temps réel (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

realtimeclock→get_hardwareId() YRealTimeClock
realtimeclock→hardwareId()

Retourne l'identifiant matériel unique de l'horloge à temps réel au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
dnp     string get_hardwareId( )
java    String get_hardwareId( )
py      get_hardwareId( )
php     function get_hardwareId( )
es      async get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'horloge à temps réel (par example RELAYLO1-123456.relay1).

Retourne :
une chaîne de caractères identifiant l'horloge à temps réel (ex: RELAYLO1-123456.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

realtimeclock→get_logicalName() YRealTimeClock
realtimeclock→logicalName()

Retourne le nom logique de l'horloge à temps réel.

```
js      function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     string get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
dnp     string get_logicalName( )
java    String get_logicalName( )
uwp     async Task<string> get_logicalName( )
py      get_logicalName( )
php     function get_logicalName( )
es      async get_logicalName( )
cmd     YRealTimeClock target get_logicalName
```

Retourne :
une chaîne de caractères représentant le nom logique de l'horloge à temps réel.

En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

realtimeclock→get_module()
realtimeclock→module()

YRealTimeClock

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function **get_module**( )
cpp	YModule * **get_module**( )
m	-(YModule*) module
pas	TYModule **get_module**( ): TYModule
vb	function **get_module**( ) As YModule
cs	YModule **get_module**( )
dnp	YModuleProxy **get_module**( )
java	YModule **get_module**( )
py	**get_module**( )
php	function **get_module**( )
es	async **get_module**( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Retourne :
une instance de YModule

realtimeclock→get_module_async() YRealTimeClock
realtimeclock→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

realtimeclock→get_serialNumber() YRealTimeClock
realtimeclock→serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

```
js      function get_serialNumber( )
cpp     string get_serialNumber( )
m       -(NSString*) serialNumber
pas     string get_serialNumber( ): string
vb      function get_serialNumber( ) As String
cs      string get_serialNumber( )
dnp     string get_serialNumber( )
java    String get_serialNumber( )
uwp     async Task<string> get_serialNumber( )
py      get_serialNumber( )
php     function get_serialNumber( )
es      async get_serialNumber( )
cmd     YRealTimeClock target get_serialNumber
```

Retourne :

: une chaîne de caractères représentant le numéro de série du module, préprogrammé en usine.

En cas d'erreur, déclenche une exception ou retourne YModule.SERIALNUMBER_INVALID.

realtimeclock→get_timeSet()
realtimeclock→timeSet()

YRealTimeClock

Retourne vrai si l'horloge à été mise à l'heure, sinon faux.

```
js    function get_timeSet( )
cpp   Y_TIMESET_enum get_timeSet( )
m     -(Y_TIMESET_enum) timeSet
pas   Integer get_timeSet( ): Integer
vb    function get_timeSet( ) As Integer
cs    int get_timeSet( )
dnp   int get_timeSet( )
java  int get_timeSet( )
uwp   async Task<int> get_timeSet( )
py    get_timeSet( )
php   function get_timeSet( )
es    async get_timeSet( )
cmd   YRealTimeClock target get_timeSet
```

Retourne :

soit `Y_TIMESET_FALSE`, soit `Y_TIMESET_TRUE`, selon vrai si l'horloge à été mise à l'heure, sinon faux

En cas d'erreur, déclenche une exception ou retourne `Y_TIMESET_INVALID`.

realtimeclock→get_unixTime()
realtimeclock→unixTime()

YRealTimeClock

Retourne l'heure courante au format Unix (nombre de seconds secondes écoulées depuis le 1er janvier 1970).

```
js      function get_unixTime( )
cpp     s64 get_unixTime( )
m       -(s64) unixTime
pas     int64 get_unixTime( ): int64
vb      function get_unixTime( ) As Long
cs      long get_unixTime( )
dnp     long get_unixTime( )
java    long get_unixTime( )
uwp     async Task<long> get_unixTime( )
py      get_unixTime( )
php     function get_unixTime( )
es      async get_unixTime( )
cmd     YRealTimeClock target get_unixTime
```

Retourne :

un entier représentant l'heure courante au format Unix (nombre de seconds secondes écoulées depuis le 1er janvier 1970)

En cas d'erreur, déclenche une exception ou retourne `Y_UNIXTIME_INVALID`.

realtimeclock→get_userData() YRealTimeClock
realtimeclock→userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

```
js      function get_userData( )
cpp     void * get_userData( )
m       -(id) userData
pas     Tobject get_userData( ): Tobject
vb      function get_userData( ) As Object
cs      object get_userData( )
java    Object get_userData( )
py      get_userData( )
php     function get_userData( )
es      async get_userData( )
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Retourne :
l'objet stocké précédemment par l'appelant.

realtimeclock→get_utcOffset()
realtimeclock→utcOffset()

YRealTimeClock

Retourne le nombre de secondes de décallage entre l'heure courante et l'heure UTC (time zone).

```
js      function get_utcOffset( )
cpp     int get_utcOffset( )
m       -(int) utcOffset
pas     LongInt get_utcOffset( ): LongInt
vb      function get_utcOffset( ) As Integer
cs      int get_utcOffset( )
dnp     int get_utcOffset( )
java    int get_utcOffset( )
uwp     async Task<int> get_utcOffset( )
py      get_utcOffset( )
php     function get_utcOffset( )
es      async get_utcOffset( )
cmd     YRealTimeClock target get_utcOffset
```

Retourne :

un entier représentant le nombre de secondes de décallage entre l'heure courante et l'heure UTC (time zone)

En cas d'erreur, déclenche une exception ou retourne `Y_UTCOFFSET_INVALID`.

realtimeclock→isOnline() YRealTimeClock

Vérifie si le module hébergeant l'horloge à temps réel est joignable, sans déclencher d'erreur.

js	function **isOnline**( )
cpp	bool **isOnline**( )
m	-(BOOL) **isOnline**
pas	boolean **isOnline**( ): boolean
vb	function **isOnline**( ) As Boolean
cs	bool **isOnline**( )
dnp	bool **isOnline**( )
java	boolean **isOnline**( )
py	**isOnline**( )
php	function **isOnline**( )
es	async **isOnline**( )

Si les valeurs des attributs en cache de l'horloge à temps réel sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Retourne :
`true` si l'horloge à temps réel est joignable, `false` sinon

realtimeclock→isOnline_async() YRealTimeClock

Vérifie si le module hébergeant l'horloge à temps réel est joignable, sans déclencher d'erreur.

```
js  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache de l'horloge à temps réel sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

realtimeclock→isReadOnly() YRealTimeClock

Test si la fonction est en lecture seule.

cpp	bool **isReadOnly**( )
m	-(bool) **isReadOnly**
pas	boolean **isReadOnly**( ): boolean
vb	function **isReadOnly**( ) As Boolean
cs	bool **isReadOnly**( )
dnp	bool **isReadOnly**( )
java	boolean **isReadOnly**( )
uwp	async Task<bool> **isReadOnly**( )
py	**isReadOnly**( )
php	function **isReadOnly**( )
es	async **isReadOnly**( )
cmd	YRealTimeClock **target isReadOnly**

Retourne vrais si la fonction est protégé en ecriture ou que la fontion n'est pas disponible.

Retourne :
`true` si la fonction est protégé en ecriture ou que la fontion n'est pas disponible

realtimeclock→load() YRealTimeClock

Met en cache les valeurs courantes de l'horloge à temps réel, avec une durée de validité spécifiée.

```
js      function load( msValidity)
cpp     YRETCODE load( int msValidity)
m       -(YRETCODE) load : (u64) msValidity
pas     YRETCODE load( msValidity: u64): YRETCODE
vb      function load( ByVal msValidity As Long) As YRETCODE
cs      YRETCODE load( ulong msValidity)
java    int load( long msValidity)
py      load( msValidity)
php     function load( $msValidity)
es      async load( msValidity)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

realtimeclock→loadAttribute() YRealTimeClock

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

```
js      function loadAttribute( attrName)
cpp     string loadAttribute( string attrName)
m       -(NSString*) loadAttribute : (NSString*) attrName
pas     string loadAttribute( attrName: string): string
vb      function loadAttribute( ) As String
cs      string loadAttribute( string attrName)
dnp     string loadAttribute( string attrName)
java    String loadAttribute( String attrName)
uwp     async Task<string> loadAttribute( string attrName)
py      loadAttribute( attrName)
php     function loadAttribute( $attrName)
es      async loadAttribute( attrName)
```

Paramètres :

attrName le nom de l'attribut désiré

Retourne :

une chaîne de caractères représentant la valeur actuelle de l'attribut.

En cas d'erreur, déclenche une exception ou retourne un chaîne vide.

realtimeclock→load_async() YRealTimeClock

Met en cache les valeurs courantes de l'horloge à temps réel, avec une durée de validité spécifiée.

```
js  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

msValidity	un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
callback	fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)
context	contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

realtimeclock→muteValueCallbacks() YRealTimeClock

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

js	function **muteValueCallbacks**( )
cpp	int **muteValueCallbacks**( )
m	-(int) **muteValueCallbacks**
pas	LongInt **muteValueCallbacks**( ): LongInt
vb	function **muteValueCallbacks**( ) As Integer
cs	int **muteValueCallbacks**( )
dnp	int **muteValueCallbacks**( )
java	int **muteValueCallbacks**( )
uwp	async Task<int> **muteValueCallbacks**( )
py	**muteValueCallbacks**( )
php	function **muteValueCallbacks**( )
es	async **muteValueCallbacks**( )
cmd	YRealTimeClock **target** **muteValueCallbacks**

Vous pouvez utiliser cette fonction pour économiser la bande passante et le CPU sur les machines de faible puissance, ou pour éviter le déclanchement de callbacks HTTP. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

realtimeclock→nextRealTimeClock() YRealTimeClock

Continue l'énumération des horloges à temps réel commencée à l'aide de `yFirstRealTimeClock()` Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les horloges à temps réel sont retournés.

js	function **nextRealTimeClock**( )
cpp	YRealTimeClock * **nextRealTimeClock**( )
m	-(YRealTimeClock*) **nextRealTimeClock**
pas	TYRealTimeClock **nextRealTimeClock**( ): TYRealTimeClock
vb	function **nextRealTimeClock**( ) As YRealTimeClock
cs	YRealTimeClock **nextRealTimeClock**( )
java	YRealTimeClock **nextRealTimeClock**( )
uwp	YRealTimeClock **nextRealTimeClock**( )
py	**nextRealTimeClock**( )
php	function **nextRealTimeClock**( )
es	**nextRealTimeClock**( )

Si vous souhaitez retrouver une horloge à temps réel spécifique, utilisez `RealTimeClock.findRealTimeClock()` avec un hardwareID ou un nom logique.

Retourne :

un pointeur sur un objet `YRealTimeClock` accessible en ligne, ou `null` lorsque l'énumération est terminée.

realtimeclock→registerValueCallback() YRealTimeClock

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

```
js    function registerValueCallback( callback)
cpp   int registerValueCallback( YRealTimeClockValueCallback callback)
m     -(int) registerValueCallback : (YRealTimeClockValueCallback) callback
pas   LongInt registerValueCallback( callback: TYRealTimeClockValueCallback): LongInt
vb    function registerValueCallback( ) As Integer
cs    int registerValueCallback( ValueCallback callback)
java  int registerValueCallback( UpdateCallback callback)
uwp   async Task<int> registerValueCallback( ValueCallback callback)
py    registerValueCallback( callback)
php   function registerValueCallback( $callback)
es    async registerValueCallback( callback)
```

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

Paramètres :

callback la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

realtimeclock→set_logicalName() YRealTimeClock
realtimeclock→setLogicalName()

Modifie le nom logique de l'horloge à temps réel.

js	function **set_logicalName**(**newval**)
cpp	int **set_logicalName**(string **newval**)
m	-(int) setLogicalName : (NSString*) **newval**
pas	integer **set_logicalName**(**newval**: string): integer
vb	function **set_logicalName**(ByVal **newval** As String) As Integer
cs	int **set_logicalName**(string **newval**)
dnp	int **set_logicalName**(string **newval**)
java	int **set_logicalName**(String **newval**)
uwp	async Task<int> **set_logicalName**(string **newval**)
py	**set_logicalName**(**newval**)
php	function **set_logicalName**(**$newval**)
es	async **set_logicalName**(**newval**)
cmd	YRealTimeClock **target set_logicalName newval**

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom logique de l'horloge à temps réel.

Retourne :

`YAPI_SUCCESS` si l'appel se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

realtimeclock→set_unixTime()
realtimeclock→setUnixTime()

YRealTimeClock

Modifie l'heure courante.

```
js      function set_unixTime( newval)
cpp     int set_unixTime( s64 newval)
m       -(int) setUnixTime : (s64) newval
pas     integer set_unixTime( newval: int64): integer
vb      function set_unixTime( ByVal newval As Long) As Integer
cs      int set_unixTime( long newval)
dnp     int set_unixTime( long newval)
java    int set_unixTime( long newval)
uwp     async Task<int> set_unixTime( long newval)
py      set_unixTime( newval)
php     function set_unixTime( $newval)
es      async set_unixTime( newval)
cmd     YRealTimeClock target set_unixTime newval
```

L'heure est passée au format Unix (nombre de seconds secondes écoulées depuis le 1er janvier 1970).

Paramètres :

newval un entier représentant l'heure courante

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

realtimeclock→set_userData() YRealTimeClock
realtimeclock→setUserData()

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js    function set_userData( data)
cpp   void set_userData( void * data)
m     -(void) setUserData : (id) data
pas   set_userData( data: Tobject)
vb    procedure set_userData( ByVal data As Object)
cs    void set_userData( object data)
java  void set_userData( Object data)
py    set_userData( data)
php   function set_userData( $data)
es    async set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Paramètres :

data objet quelconque à mémoriser

realtimeclock→set_utcOffset() YRealTimeClock
realtimeclock→setUtcOffset()

Modifie le nombre de secondes de décalage entre l'heure courante et l'heure UTC (time zone).

```
js    function set_utcOffset( newval)
cpp   int set_utcOffset( int newval)
m     -(int) setUtcOffset : (int) newval
pas   integer set_utcOffset( newval: LongInt): integer
vb    function set_utcOffset( ByVal newval As Integer) As Integer
cs    int set_utcOffset( int newval)
dnp   int set_utcOffset( int newval)
java  int set_utcOffset( int newval)
uwp   async Task<int> set_utcOffset( int newval)
py    set_utcOffset( newval)
php   function set_utcOffset( $newval)
es    async set_utcOffset( newval)
cmd   YRealTimeClock target set_utcOffset newval
```

Le décallage est automatiquement arrondi au quart d'heure le plus proche. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant le nombre de secondes de décalage entre l'heure courante et l'heure UTC (time zone)

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

realtimeclock→unmuteValueCallbacks() YRealTimeClock

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

js	function **unmuteValueCallbacks**( )
cpp	int **unmuteValueCallbacks**( )
m	-(int) **unmuteValueCallbacks**
pas	LongInt **unmuteValueCallbacks**( ): LongInt
vb	function **unmuteValueCallbacks**( ) As Integer
cs	int **unmuteValueCallbacks**( )
dnp	int **unmuteValueCallbacks**( )
java	int **unmuteValueCallbacks**( )
uwp	async Task<int> **unmuteValueCallbacks**( )
py	**unmuteValueCallbacks**( )
php	function **unmuteValueCallbacks**( )
es	async **unmuteValueCallbacks**( )
cmd	YRealTimeClock **target** **unmuteValueCallbacks**

Cette fonction annule un précédent appel à `muteValueCallbacks()`. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

realtimeclock→wait_async() YRealTimeClock

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js  function wait_async( callback, context)
es  wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

Paramètres :

callback fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout.

8.6. La classe YWakeUpMonitor

Interface pour intéragir avec les moniteurs de réveil, disponibles par exemple dans le YoctoHub-GSM-3G-EU, le YoctoHub-GSM-3G-NA, le YoctoHub-Wireless-SR et le YoctoHub-Wireless-g

La classe `YWakeUpMonitor` prend en charge le contrôle global de toutes les sources de réveil possibles ainsi que les mises en sommeil automatiques.

Pour utiliser les fonctions décrites ici, vous devez inclure:

es	in HTML: <script src="../../lib/yocto_wakeupmonitor.js"></script> in node.js: require('yoctolib-es2017/yocto_wakeupmonitor.js');
js	<script type='text/javascript' src='yocto_wakeupmonitor.js'></script>
cpp	#include "yocto_wakeupmonitor.h"
m	#import "yocto_wakeupmonitor.h"
pas	uses yocto_wakeupmonitor;
vb	yocto_wakeupmonitor.vb
cs	yocto_wakeupmonitor.cs
dnp	import YoctoProxyAPI.YWakeUpMonitorProxy
java	import com.yoctopuce.YoctoAPI.YWakeUpMonitor;
uwp	import com.yoctopuce.YoctoAPI.YWakeUpMonitor;
py	from yocto_wakeupmonitor import *
php	require_once('yocto_wakeupmonitor.php');
vi	YWakeUpMonitor.vi

Fonction globales

YWakeUpMonitor.FindWakeUpMonitor(**func**)

Permet de retrouver un moniteur de réveil d'après un identifiant donné.

YWakeUpMonitor.FindWakeUpMonitorInContext(**yctx**, **func**)

Permet de retrouver un moniteur de réveil d'après un identifiant donné dans un Context YAPI.

YWakeUpMonitor.FirstWakeUpMonitor()

Commence l'énumération des moniteurs de réveil accessibles par la librairie.

YWakeUpMonitor.FirstWakeUpMonitorInContext(**yctx**)

Commence l'énumération des moniteurs de réveil accessibles par la librairie.

YWakeUpMonitor.GetSimilarFunctions()

Enumère toutes les fonctions de type WakeUpMonitor disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).

Propriétés des objets `YWakeUpMonitorProxy`

wakeupmonitor→AdvertisedValue *[lecture seule]*

Courte chaîne de caractères représentant l'état courant de la fonction.

wakeupmonitor→FriendlyName *[lecture seule]*

Identifiant global de la fonction au format `NOM_MODULE.NOM_FONCTION`.

wakeupmonitor→FunctionId *[lecture seule]*

Identifiant matériel du moniteur de réveil, sans référence au module.

wakeupmonitor→HardwareId *[lecture seule]*

Identifiant matériel unique de la fonction au format `SERIAL.FUNCTIONID`.

wakeupmonitor→IsOnline *[lecture seule]*

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

wakeupmonitor→LogicalName *[modifiable]*

Nom logique de la fonction.

wakeupmonitor→NextWakeUp *[modifiable]*

Prochaine date/heure de réveil agendée (format UNIX).

wakeupmonitor→PowerDuration *[modifiable]*

Temp d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

wakeupmonitor→SerialNumber *[lecture seule]*

Numéro de série du module, préprogrammé en usine.

Méthodes des objets YWakeUpMonitor

wakeupmonitor→clearCache()

Invalide le cache.

wakeupmonitor→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du moniteur de réveil au format TYPE(NAME)=SERIAL.FUNCTIONID.

wakeupmonitor→get_advertisedValue()

Retourne la valeur courante du moniteur de réveil (pas plus de 6 caractères).

wakeupmonitor→get_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du moniteur de réveil.

wakeupmonitor→get_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du moniteur de réveil.

wakeupmonitor→get_friendlyName()

Retourne un identifiant global du moniteur de réveil au format NOM_MODULE.NOM_FONCTION.

wakeupmonitor→get_functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

wakeupmonitor→get_functionId()

Retourne l'identifiant matériel du moniteur de réveil, sans référence au module.

wakeupmonitor→get_hardwareId()

Retourne l'identifiant matériel unique du moniteur de réveil au format SERIAL.FUNCTIONID.

wakeupmonitor→get_logicalName()

Retourne le nom logique du moniteur de réveil.

wakeupmonitor→get_module()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

wakeupmonitor→get_module_async(**callback**, **context**)

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

wakeupmonitor→get_nextWakeUp()

Retourne la prochaine date/heure de réveil agendée (format UNIX).

wakeupmonitor→get_powerDuration()

Retourne le temp d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

wakeupmonitor→get_serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

wakeupmonitor→get_sleepCountdown()

Retourne le temps avant le prochain sommeil.

wakeupmonitor→get_userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set_userData.

wakeupmonitor→get_wakeUpReason()

Renvoie la raison du dernier réveil.

wakeupmonitor→get_wakeUpState()

Revoie l'état actuel du moniteur.

wakeupmonitor→isOnline()

Vérifie si le module hébergeant le moniteur de réveil est joignable, sans déclencher d'erreur.

wakeupmonitor→isOnline_async(**callback**, **context**)

Vérifie si le module hébergeant le moniteur de réveil est joignable, sans déclencher d'erreur.

wakeupmonitor→isReadOnly()

Test si la fonction est en lecture seule.

wakeupmonitor→load(**msValidity**)

Met en cache les valeurs courantes du moniteur de réveil, avec une durée de validité spécifiée.

wakeupmonitor→loadAttribute(**attrName**)

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

wakeupmonitor→load_async(**msValidity**, **callback**, **context**)

Met en cache les valeurs courantes du moniteur de réveil, avec une durée de validité spécifiée.

wakeupmonitor→muteValueCallbacks()

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

wakeupmonitor→nextWakeUpMonitor()

Continue l'énumération des moniteurs de réveil commencée à l'aide de `yFirstWakeUpMonitor()` Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les moniteurs de réveil sont retournés.

wakeupmonitor→registerValueCallback(**callback**)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

wakeupmonitor→resetSleepCountDown()

Réinitialise le compteur de mise en sommeil.

wakeupmonitor→set_logicalName(**newval**)

Modifie le nom logique du moniteur de réveil.

wakeupmonitor→set_nextWakeUp(**newval**)

Modifie les jours de la semaine où un réveil doit avoir lieu.

wakeupmonitor→set_powerDuration(**newval**)

Modifie le temps d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

wakeupmonitor→set_sleepCountdown(**newval**)

Modifie le temps avant le prochain sommeil .

wakeupmonitor→set_userData(**data**)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

wakeupmonitor→sleep(**secBeforeSleep**)

Déclenche une mise en sommeil jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

wakeupmonitor→sleepFor(**secUntilWakeUp**, **secBeforeSleep**)

Déclenche une mise en sommeil pour un temps donné ou jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

wakeupmonitor→sleepUntil(**wakeUpTime**, **secBeforeSleep**)

Déclenche une mise en sommeil jusqu'à une date donnée ou jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

wakeupmonitor→unmuteValueCallbacks()

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

wakeupmonitor→wait_async(callback, context)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

wakeupmonitor→wakeUp()

Force un réveil.

YWakeUpMonitor.FindWakeUpMonitor() YWakeUpMonitor
YWakeUpMonitor.FindWakeUpMonitor()

Permet de retrouver un moniteur de réveil d'après un identifiant donné.

```
js      function yFindWakeUpMonitor( func)
cpp     YWakeUpMonitor* yFindWakeUpMonitor( string func)
m       +(YWakeUpMonitor*) FindWakeUpMonitor : (NSString*) func
pas     TYWakeUpMonitor yFindWakeUpMonitor( func: string): TYWakeUpMonitor
vb      function yFindWakeUpMonitor( ByVal func As String) As YWakeUpMonitor
cs      static YWakeUpMonitor FindWakeUpMonitor( string func)
dnp     static YWakeUpMonitorProxy FindWakeUpMonitor( string func)
java    static YWakeUpMonitor FindWakeUpMonitor( String func)
uwp     static YWakeUpMonitor FindWakeUpMonitor( string func)
py      FindWakeUpMonitor( func)
php     function yFindWakeUpMonitor( $func)
es      static FindWakeUpMonitor( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le moniteur de réveil soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YWakeUpMonitor.isOnline()` pour tester si le moniteur de réveil est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Si un appel à la méthode is_online() de cet objet renvoie FAUX alors que vous êtes sûr que le module correspondant est bien branché, vérifiez que vous n'avez pas oublié d'appeler registerHub() à l'initialisation de de l'application.

Paramètres :

func une chaîne de caractères qui référence le moniteur de réveil sans ambiguïté, par exemple `YHUBGSM3.wakeUpMonitor`.

Retourne :

un objet de classe `YWakeUpMonitor` qui permet ensuite de contrôler le moniteur de réveil.

YWakeUpMonitor.FindWakeUpMonitorInContext() YWakeUpMonitor
YWakeUpMonitor.FindWakeUpMonitorInContext()

Permet de retrouver un moniteur de réveil d'après un identifiant donné dans un Context YAPI.

```
java  static YWakeUpMonitor FindWakeUpMonitorInContext( YAPIContext yctx,
                                                        String func)
uwp   static YWakeUpMonitor FindWakeUpMonitorInContext( YAPIContext yctx,
                                                        string func)
es    static FindWakeUpMonitorInContext( yctx, func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le moniteur de réveil soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YWakeUpMonitor.isOnline()` pour tester si le moniteur de réveil est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Paramètres :

yctx un contexte YAPI

func une chaîne de caractères qui référence le moniteur de réveil sans ambiguïté, par exemple `YHUBGSM3.wakeUpMonitor`.

Retourne :

un objet de classe `YWakeUpMonitor` qui permet ensuite de contrôler le moniteur de réveil.

YWakeUpMonitor.FirstWakeUpMonitor() YWakeUpMonitor

YWakeUpMonitor.FirstWakeUpMonitor()

Commence l'énumération des moniteurs de réveil accessibles par la librairie.

js	function **yFirstWakeUpMonitor**( )
cpp	YWakeUpMonitor * **yFirstWakeUpMonitor**( )
m	+(YWakeUpMonitor*) **FirstWakeUpMonitor**
pas	TYWakeUpMonitor **yFirstWakeUpMonitor**( ): TYWakeUpMonitor
vb	function **yFirstWakeUpMonitor**( ) As YWakeUpMonitor
cs	static YWakeUpMonitor **FirstWakeUpMonitor**( )
java	static YWakeUpMonitor **FirstWakeUpMonitor**( )
uwp	static YWakeUpMonitor **FirstWakeUpMonitor**( )
py	**FirstWakeUpMonitor**( )
php	function **yFirstWakeUpMonitor**( )
es	static **FirstWakeUpMonitor**( )

Utiliser la fonction `YWakeUpMonitor.nextWakeUpMonitor()` pour itérer sur les autres moniteurs de réveil.

Retourne :

un pointeur sur un objet `YWakeUpMonitor`, correspondant au premier moniteur de réveil accessible en ligne, ou `null` si il n'y a pas de moniteurs de réveil disponibles.

YWakeUpMonitor.FirstWakeUpMonitorInContext() YWakeUpMonitor YWakeUpMonitor.FirstWakeUpMonitorInContext()

Commence l'énumération des moniteurs de réveil accessibles par la librairie.

```
java  static YWakeUpMonitor FirstWakeUpMonitorInContext( YAPIContext yctx)
uwp   static YWakeUpMonitor FirstWakeUpMonitorInContext( YAPIContext yctx)
es    static FirstWakeUpMonitorInContext( yctx)
```

Utiliser la fonction `YWakeUpMonitor.nextWakeUpMonitor()` pour itérer sur les autres moniteurs de réveil.

Paramètres :

yctx un contexte YAPI.

Retourne :

un pointeur sur un objet `YWakeUpMonitor`, correspondant au premier moniteur de réveil accessible en ligne, ou `null` si il n'y a pas de moniteurs de réveil disponibles.

YWakeUpMonitor.GetSimilarFunctions() YWakeUpMonitor YWakeUpMonitor.GetSimilarFunctions()

Enumère toutes les fonctions de type WakeUpMonitor disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).

dnp static new string[] **GetSimilarFunctions**( )

Chaque chaîne retournée peut être passée en argument à la méthode `YWakeUpMonitor.FindWakeUpMonitor` pour obtenir une objet permettant d'intéragir avec le module correspondant.

Retourne :

un tableau de chaînes de caractères, contenant les identifiants matériels de chaque fonction disponible trouvée.

wakeupmonitor→AdvertisedValue — YWakeUpMonitor

Courte chaîne de caractères représentant l'état courant de la fonction.

dnp string **AdvertisedValue**

wakeupmonitor→FriendlyName — YWakeUpMonitor

Identifiant global de la fonction au format `NOM_MODULE.NOM_FONCTION`.

dnp string **FriendlyName**

Le chaîne retournée utilise soit les noms logiques du module et de la fonction si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel de la fonction (par exemple: `MyCustomName.relay1`)

wakeupmonitor→FunctionId — YWakeUpMonitor

Identifiant matériel du moniteur de réveil, sans référence au module.

dnp string **FunctionId**

Par example `relay1`.

wakeupmonitor→HardwareId — YWakeUpMonitor

Identifiant matériel unique de la fonction au format SERIAL.FUNCTIONID.

dnp string **HardwareId**

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de la fonction (par example RELAYLO1-123456.relay1).

wakeupmonitor→IsOnline — YWakeUpMonitor

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

dnp bool **IsOnline**

Si les valeurs des attributs en cache de la fonction sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

wakeupmonitor→LogicalName YWakeUpMonitor

Nom logique de la fonction.

dnp string **LogicalName**

Modifiable. Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

wakeupmonitor→NextWakeUp — YWakeUpMonitor

Prochaine date/heure de réveil agendée (format UNIX).

dnp long **NextWakeUp**

Modifiable. Modifie les jours de la semaine où un réveil doit avoir lieu.

wakeupmonitor→PowerDuration YWakeUpMonitor

Temp d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

dnp int **PowerDuration**

Modifiable. Modifie le temps d'éveil maximal en secondes avant de retourner en sommeil automatiquement. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

wakeupmonitor→SerialNumber — YWakeUpMonitor

Numéro de série du module, préprogrammé en usine.

dnp string **SerialNumber**

wakeupmonitor→clearCache() YWakeUpMonitor

Invalide le cache.

js	function **clearCache**( )
cpp	void **clearCache**( )
m	-(void) **clearCache**
pas	**clearCache**( )
vb	procedure **clearCache**( )
cs	void **clearCache**( )
java	void **clearCache**( )
py	**clearCache**( )
php	function **clearCache**( )
es	async **clearCache**( )

Invalide le cache des valeurs courantes du moniteur de réveil. Force le prochain appel à une méthode get_xxx() ou loadxxx() pour charger les les données depuis le module.

wakeupmonitor→describe() YWakeUpMonitor

Retourne un court texte décrivant de manière non-ambigüe l'instance du moniteur de réveil au format TYPE(NAME)=SERIAL.FUNCTIONID.

```
js      function describe( )
cpp     string describe( )
m       -(NSString*) describe
pas     string describe( ): string
vb      function describe( ) As String
cs      string describe( )
java    String describe( )
py      describe( )
php     function describe( )
es      async describe( )
```

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

Retourne :
une chaîne de caractères décrivant le moniteur de réveil (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

wakeupmonitor→get_advertisedValue() YWakeUpMonitor
wakeupmonitor→advertisedValue()

Retourne la valeur courante du moniteur de réveil (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     string get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
dnp     string get_advertisedValue( )
java    String get_advertisedValue( )
uwp     async Task<string> get_advertisedValue( )
py      get_advertisedValue( )
php     function get_advertisedValue( )
es      async get_advertisedValue( )
cmd     YWakeUpMonitor target get_advertisedValue
```

Retourne :
une chaîne de caractères représentant la valeur courante du moniteur de réveil (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

wakeupmonitor→get_errorMessage() YWakeUpMonitor
wakeupmonitor→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du moniteur de réveil.

```
js      function get_errorMessage( )
cpp     string get_errorMessage( )
m       -(NSString*) errorMessage
pas     string get_errorMessage( ): string
vb      function get_errorMessage( ) As String
cs      string get_errorMessage( )
java    String get_errorMessage( )
py      get_errorMessage( )
php     function get_errorMessage( )
es      get_errorMessage( )
```

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :

une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du moniteur de réveil.

wakeupmonitor→get_errorType() YWakeUpMonitor
wakeupmonitor→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du moniteur de réveil.

js	function **get_errorType**( )
cpp	YRETCODE **get_errorType**( )
m	-(YRETCODE) errorType
pas	YRETCODE **get_errorType**( ): YRETCODE
vb	function **get_errorType**( ) As YRETCODE
cs	YRETCODE **get_errorType**( )
java	int **get_errorType**( )
py	**get_errorType**( )
php	function **get_errorType**( )
es	**get_errorType**( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du moniteur de réveil.

wakeupmonitor→get_friendlyName() YWakeUpMonitor
wakeupmonitor→friendlyName()

Retourne un identifiant global du moniteur de réveil au format NOM_MODULE.NOM_FONCTION.

```
js      function get_friendlyName( )
cpp     string get_friendlyName( )
m       -(NSString*) friendlyName
cs      string get_friendlyName( )
dnp     string get_friendlyName( )
java    String get_friendlyName( )
py      get_friendlyName( )
php     function get_friendlyName( )
es      async get_friendlyName( )
```

Le chaîne retournée utilise soit les noms logiques du module et du moniteur de réveil si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel du moniteur de réveil (par exemple: MyCustomName.relay1)

Retourne :

une chaîne de caractères identifiant le moniteur de réveil en utilisant les noms logiques (ex: MyCustomName.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

wakeupmonitor→get_functionDescriptor() YWakeUpMonitor
wakeupmonitor→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function **get_functionDescriptor**( )
cpp	YFUN_DESCR **get_functionDescriptor**( )
m	-(YFUN_DESCR) functionDescriptor
pas	YFUN_DESCR **get_functionDescriptor**( ): YFUN_DESCR
vb	function **get_functionDescriptor**( ) As YFUN_DESCR
cs	YFUN_DESCR **get_functionDescriptor**( )
java	String **get_functionDescriptor**( )
py	**get_functionDescriptor**( )
php	function **get_functionDescriptor**( )
es	async **get_functionDescriptor**( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

Retourne :
un identifiant de type YFUN_DESCR.

Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

wakeupmonitor→get_functionId()
wakeupmonitor→functionId()

YWakeUpMonitor

Retourne l'identifiant matériel du moniteur de réveil, sans référence au module.

```
js      function get_functionId( )
cpp     string get_functionId( )
m       -(NSString*) functionId
vb      function get_functionId( ) As String
cs      string get_functionId( )
dnp     string get_functionId( )
java    String get_functionId( )
py      get_functionId( )
php     function get_functionId( )
es      async get_functionId( )
```

Par example `relay1`.

Retourne :
une chaîne de caractères identifiant le moniteur de réveil (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

wakeupmonitor→get_hardwareId() YWakeUpMonitor
wakeupmonitor→hardwareId()

Retourne l'identifiant matériel unique du moniteur de réveil au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
dnp     string get_hardwareId( )
java    String get_hardwareId( )
py      get_hardwareId( )
php     function get_hardwareId( )
es      async get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du moniteur de réveil (par example RELAYLO1-123456.relay1).

Retourne :
une chaîne de caractères identifiant le moniteur de réveil (ex: RELAYLO1-123456.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

wakeupmonitor→get_logicalName() YWakeUpMonitor
wakeupmonitor→logicalName()

Retourne le nom logique du moniteur de réveil.

```
js      function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     string get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
dnp     string get_logicalName( )
java    String get_logicalName( )
uwp     async Task<string> get_logicalName( )
py      get_logicalName( )
php     function get_logicalName( )
es      async get_logicalName( )
cmd     YWakeUpMonitor target get_logicalName
```

Retourne :
une chaîne de caractères représentant le nom logique du moniteur de réveil.

En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

wakeupmonitor→get_module()
wakeupmonitor→module()

YWakeUpMonitor

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

js	function **get_module**( )
cpp	YModule * **get_module**( )
m	-(YModule*) module
pas	TYModule **get_module**( ): TYModule
vb	function **get_module**( ) As YModule
cs	YModule **get_module**( )
dnp	YModuleProxy **get_module**( )
java	YModule **get_module**( )
py	**get_module**( )
php	function **get_module**( )
es	async **get_module**( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de `YModule` retournée ne sera pas joignable.

Retourne :
une instance de `YModule`

wakeupmonitor→get_module_async() YWakeUpMonitor
wakeupmonitor→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wakeupmonitor→get_nextWakeUp() YWakeUpMonitor
wakeupmonitor→nextWakeUp()

Retourne la prochaine date/heure de réveil agendée (format UNIX).

```
js      function get_nextWakeUp( )
cpp     s64 get_nextWakeUp( )
m       -(s64) nextWakeUp
pas     int64 get_nextWakeUp( ): int64
vb      function get_nextWakeUp( ) As Long
cs      long get_nextWakeUp( )
dnp     long get_nextWakeUp( )
java    long get_nextWakeUp( )
uwp     async Task<long> get_nextWakeUp( )
py      get_nextWakeUp( )
php     function get_nextWakeUp( )
es      async get_nextWakeUp( )
cmd     YWakeUpMonitor target get_nextWakeUp
```

Retourne :

un entier représentant la prochaine date/heure de réveil agendée (format UNIX)

En cas d'erreur, déclenche une exception ou retourne `Y_NEXTWAKEUP_INVALID`.

wakeupmonitor→get_powerDuration() YWakeUpMonitor
wakeupmonitor→powerDuration()

Retourne le temp d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

```
js      function get_powerDuration( )
cpp     int get_powerDuration( )
m       -(int) powerDuration
pas     LongInt get_powerDuration( ): LongInt
vb      function get_powerDuration( ) As Integer
cs      int get_powerDuration( )
dnp     int get_powerDuration( )
java    int get_powerDuration( )
uwp     async Task<int> get_powerDuration( )
py      get_powerDuration( )
php     function get_powerDuration( )
es      async get_powerDuration( )
cmd     YWakeUpMonitor target get_powerDuration
```

Retourne :

un entier représentant le temp d'éveil maximal en secondes avant de retourner en sommeil automatiquement

En cas d'erreur, déclenche une exception ou retourne `Y_POWERDURATION_INVALID`.

wakeupmonitor→get_serialNumber() YWakeUpMonitor
wakeupmonitor→serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

```
js      function get_serialNumber( )
cpp     string get_serialNumber( )
m       -(NSString*) serialNumber
pas     string get_serialNumber( ): string
vb      function get_serialNumber( ) As String
cs      string get_serialNumber( )
dnp     string get_serialNumber( )
java    String get_serialNumber( )
uwp     async Task<string> get_serialNumber( )
py      get_serialNumber( )
php     function get_serialNumber( )
es      async get_serialNumber( )
cmd     YWakeUpMonitor target get_serialNumber
```

Retourne :

: une chaîne de caractères représentant le numéro de série du module, préprogrammé en usine.

En cas d'erreur, déclenche une exception ou retourne YModule.SERIALNUMBER_INVALID.

wakeupmonitor→get_sleepCountdown() YWakeUpMonitor
wakeupmonitor→sleepCountdown()

Retourne le temps avant le prochain sommeil.

```
js      function get_sleepCountdown( )
cpp     int get_sleepCountdown( )
m       -(int) sleepCountdown
pas     LongInt get_sleepCountdown( ): LongInt
vb      function get_sleepCountdown( ) As Integer
cs      int get_sleepCountdown( )
dnp     int get_sleepCountdown( )
java    int get_sleepCountdown( )
uwp     async Task<int> get_sleepCountdown( )
py      get_sleepCountdown( )
php     function get_sleepCountdown( )
es      async get_sleepCountdown( )
cmd     YWakeUpMonitor target get_sleepCountdown
```

Retourne :
un entier représentant le temps avant le prochain sommeil

En cas d'erreur, déclenche une exception ou retourne `Y_SLEEPCOUNTDOWN_INVALID`.

wakeupmonitor→get_userData() YWakeUpMonitor
wakeupmonitor→userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

js	function **get_userData**( )
cpp	void * **get_userData**( )
m	-(id) userData
pas	Tobject **get_userData**( ): Tobject
vb	function **get_userData**( ) As Object
cs	object **get_userData**( )
java	Object **get_userData**( )
py	**get_userData**( )
php	function **get_userData**( )
es	async **get_userData**( )

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Retourne :
l'objet stocké précédemment par l'appelant.

wakeupmonitor→get_wakeUpReason() YWakeUpMonitor
wakeupmonitor→wakeUpReason()

Renvoie la raison du dernier réveil.

js	function **get_wakeUpReason**( )
cpp	Y_WAKEUPREASON_enum **get_wakeUpReason**( )
m	-(Y_WAKEUPREASON_enum) wakeUpReason
pas	Integer **get_wakeUpReason**( ): Integer
vb	function **get_wakeUpReason**( ) As Integer
cs	int **get_wakeUpReason**( )
dnp	int **get_wakeUpReason**( )
java	int **get_wakeUpReason**( )
uwp	async Task<int> **get_wakeUpReason**( )
py	**get_wakeUpReason**( )
php	function **get_wakeUpReason**( )
es	async **get_wakeUpReason**( )
cmd	YWakeUpMonitor **target get_wakeUpReason**

Retourne :

une valeur parmi `Y_WAKEUPREASON_USBPOWER`, `Y_WAKEUPREASON_EXTPOWER`, `Y_WAKEUPREASON_ENDOFSLEEP`, `Y_WAKEUPREASON_EXTSIG1`, `Y_WAKEUPREASON_SCHEDULE1` et `Y_WAKEUPREASON_SCHEDULE2`

En cas d'erreur, déclenche une exception ou retourne `Y_WAKEUPREASON_INVALID`.

wakeupmonitor→get_wakeUpState() YWakeUpMonitor
wakeupmonitor→wakeUpState()

Revoie l'état actuel du moniteur.

js	function **get_wakeUpState**( )
cpp	Y_WAKEUPSTATE_enum **get_wakeUpState**( )
m	-(Y_WAKEUPSTATE_enum) wakeUpState
pas	Integer **get_wakeUpState**( ): Integer
vb	function **get_wakeUpState**( ) As Integer
cs	int **get_wakeUpState**( )
dnp	int **get_wakeUpState**( )
java	int **get_wakeUpState**( )
uwp	async Task<int> **get_wakeUpState**( )
py	**get_wakeUpState**( )
php	function **get_wakeUpState**( )
es	async **get_wakeUpState**( )
cmd	YWakeUpMonitor **target get_wakeUpState**

Retourne :
soit `Y_WAKEUPSTATE_SLEEPING`, soit `Y_WAKEUPSTATE_AWAKE`

En cas d'erreur, déclenche une exception ou retourne `Y_WAKEUPSTATE_INVALID`.

wakeupmonitor→isOnline() YWakeUpMonitor

Vérifie si le module hébergeant le moniteur de réveil est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     boolean isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
dnp     bool isOnline( )
java    boolean isOnline( )
py      isOnline( )
php     function isOnline( )
es      async isOnline( )
```

Si les valeurs des attributs en cache du moniteur de réveil sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Retourne :
`true` si le moniteur de réveil est joignable, `false` sinon

wakeupmonitor→isOnline_async() YWakeUpMonitor

Vérifie si le module hébergeant le moniteur de réveil est joignable, sans déclencher d'erreur.

```
js  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du moniteur de réveil sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wakeupmonitor→isReadOnly() YWakeUpMonitor

Test si la fonction est en lecture seule.

cpp	bool **isReadOnly**( )
m	-(bool) **isReadOnly**
pas	boolean **isReadOnly**( ): boolean
vb	function **isReadOnly**( ) As Boolean
cs	bool **isReadOnly**( )
dnp	bool **isReadOnly**( )
java	boolean **isReadOnly**( )
uwp	async Task<bool> **isReadOnly**( )
py	**isReadOnly**( )
php	function **isReadOnly**( )
es	async **isReadOnly**( )
cmd	YWakeUpMonitor **target isReadOnly**

Retourne vrais si la fonction est protégé en ecriture ou que la fontion n'est pas disponible.

Retourne :
`true` si la fonction est protégé en ecriture ou que la fontion n'est pas disponible

wakeupmonitor→load() YWakeUpMonitor

Met en cache les valeurs courantes du moniteur de réveil, avec une durée de validité spécifiée.

js	function **load**(**msValidity**)
cpp	YRETCODE **load**(int **msValidity**)
m	-(YRETCODE) **load** : (u64) **msValidity**
pas	YRETCODE **load**(**msValidity**: u64): YRETCODE
vb	function **load**(ByVal **msValidity** As Long) As YRETCODE
cs	YRETCODE **load**(ulong **msValidity**)
java	int **load**(long **msValidity**)
py	**load**(**msValidity**)
php	function **load**(**$msValidity**)
es	async **load**(**msValidity**)

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→loadAttribute() YWakeUpMonitor

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

```
js    function loadAttribute( attrName)
cpp   string loadAttribute( string attrName)
m     -(NSString*) loadAttribute : (NSString*) attrName
pas   string loadAttribute( attrName: string): string
vb    function loadAttribute( ) As String
cs    string loadAttribute( string attrName)
dnp   string loadAttribute( string attrName)
java  String loadAttribute( String attrName)
uwp   async Task<string> loadAttribute( string attrName)
py    loadAttribute( attrName)
php   function loadAttribute( $attrName)
es    async loadAttribute( attrName)
```

Paramètres :

attrName le nom de l'attribut désiré

Retourne :

une chaîne de caractères représentant la valeur actuelle de l'attribut.

En cas d'erreur, déclenche une exception ou retourne un chaîne vide.

wakeupmonitor→load_async() YWakeUpMonitor

Met en cache les valeurs courantes du moniteur de réveil, avec une durée de validité spécifiée.

```
js  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

msValidity	un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
callback	fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)
context	contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wakeupmonitor→muteValueCallbacks() YWakeUpMonitor

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

```
js      function muteValueCallbacks( )
cpp     int muteValueCallbacks( )
m       -(int) muteValueCallbacks
pas     LongInt muteValueCallbacks( ): LongInt
vb      function muteValueCallbacks( ) As Integer
cs      int muteValueCallbacks( )
dnp     int muteValueCallbacks( )
java    int muteValueCallbacks( )
uwp     async Task<int> muteValueCallbacks( )
py      muteValueCallbacks( )
php     function muteValueCallbacks( )
es      async muteValueCallbacks( )
cmd     YWakeUpMonitor target muteValueCallbacks
```

Vous pouvez utiliser cette fonction pour économiser la bande passante et le CPU sur les machines de faible puissance, ou pour éviter le déclanchement de callbacks HTTP. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→nextWakeUpMonitor() YWakeUpMonitor

Continue l'énumération des moniteurs de réveil commencée à l'aide de `yFirstWakeUpMonitor()` Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les moniteurs de réveil sont retournés.

`js`	function **nextWakeUpMonitor**( )
`cpp`	YWakeUpMonitor * **nextWakeUpMonitor**( )
`m`	-(YWakeUpMonitor*) **nextWakeUpMonitor**
`pas`	TYWakeUpMonitor **nextWakeUpMonitor**( ): TYWakeUpMonitor
`vb`	function **nextWakeUpMonitor**( ) As YWakeUpMonitor
`cs`	YWakeUpMonitor **nextWakeUpMonitor**( )
`java`	YWakeUpMonitor **nextWakeUpMonitor**( )
`uwp`	YWakeUpMonitor **nextWakeUpMonitor**( )
`py`	**nextWakeUpMonitor**( )
`php`	function **nextWakeUpMonitor**( )
`es`	**nextWakeUpMonitor**( )

Si vous souhaitez retrouver un moniteur de réveil spécifique, utilisez `WakeUpMonitor.findWakeUpMonitor()` avec un hardwareID ou un nom logique.

Retourne :

un pointeur sur un objet `YWakeUpMonitor` accessible en ligne, ou `null` lorsque l'énumération est terminée.

wakeupmonitor→registerValueCallback() YWakeUpMonitor

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

```
js      function registerValueCallback( callback)
cpp     int registerValueCallback( YWakeUpMonitorValueCallback callback)
m       -(int) registerValueCallback : (YWakeUpMonitorValueCallback) callback
pas     LongInt registerValueCallback( callback: TYWakeUpMonitorValueCallback): LongInt
vb      function registerValueCallback( ) As Integer
cs      int registerValueCallback( ValueCallback callback)
java    int registerValueCallback( UpdateCallback callback)
uwp     async Task<int> registerValueCallback( ValueCallback callback)
py      registerValueCallback( callback)
php     function registerValueCallback( $callback)
es      async registerValueCallback( callback)
```

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

Paramètres :

callback la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

wakeupmonitor→resetSleepCountDown() YWakeUpMonitor

Réinitialise le compteur de mise en sommeil.

```
js      function resetSleepCountDown( )
cpp     int resetSleepCountDown( )
m       -(int) resetSleepCountDown
pas     LongInt resetSleepCountDown( ): LongInt
vb      function resetSleepCountDown( ) As Integer
cs      int resetSleepCountDown( )
dnp     int resetSleepCountDown( )
java    int resetSleepCountDown( )
uwp     async Task<int> resetSleepCountDown( )
py      resetSleepCountDown( )
php     function resetSleepCountDown( )
es      async resetSleepCountDown( )
cmd     YWakeUpMonitor target resetSleepCountDown
```

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur. En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→set_logicalName()
wakeupmonitor→setLogicalName()

YWakeUpMonitor

Modifie le nom logique du moniteur de réveil.

```
js    function set_logicalName( newval)
cpp   int set_logicalName( string newval)
m     -(int) setLogicalName : (NSString*) newval
pas   integer set_logicalName( newval: string): integer
vb    function set_logicalName( ByVal newval As String) As Integer
cs    int set_logicalName( string newval)
dnp   int set_logicalName( string newval)
java  int set_logicalName( String newval)
uwp   async Task<int> set_logicalName( string newval)
py    set_logicalName( newval)
php   function set_logicalName( $newval)
es    async set_logicalName( newval)
cmd   YWakeUpMonitor target set_logicalName newval
```

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom logique du moniteur de réveil.

Retourne :

`YAPI_SUCCESS` si l'appel se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→set_nextWakeUp() YWakeUpMonitor
wakeupmonitor→setNextWakeUp()

Modifie les jours de la semaine où un réveil doit avoir lieu.

```
js      function set_nextWakeUp( newval)
cpp     int set_nextWakeUp( s64 newval)
m       -(int) setNextWakeUp : (s64) newval
pas     integer set_nextWakeUp( newval: int64): integer
vb      function set_nextWakeUp( ByVal newval As Long) As Integer
cs      int set_nextWakeUp( long newval)
dnp     int set_nextWakeUp( long newval)
java    int set_nextWakeUp( long newval)
uwp     async Task<int> set_nextWakeUp( long newval)
py      set_nextWakeUp( newval)
php     function set_nextWakeUp( $newval)
es      async set_nextWakeUp( newval)
cmd     YWakeUpMonitor target set_nextWakeUp newval
```

Paramètres :

newval un entier représentant les jours de la semaine où un réveil doit avoir lieu

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→set_powerDuration() YWakeUpMonitor
wakeupmonitor→setPowerDuration()

Modifie le temps d'éveil maximal en secondes avant de retourner en sommeil automatiquement.

```
js      function set_powerDuration( newval)
cpp     int set_powerDuration( int newval)
m       -(int) setPowerDuration : (int) newval
pas     integer set_powerDuration( newval: LongInt): integer
vb      function set_powerDuration( ByVal newval As Integer) As Integer
cs      int set_powerDuration( int newval)
dnp     int set_powerDuration( int newval)
java    int set_powerDuration( int newval)
uwp     async Task<int> set_powerDuration( int newval)
py      set_powerDuration( newval)
php     function set_powerDuration( $newval)
es      async set_powerDuration( newval)
cmd     YWakeUpMonitor target set_powerDuration newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant le temps d'éveil maximal en secondes avant de retourner en sommeil automatiquement

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→set_sleepCountdown() YWakeUpMonitor
wakeupmonitor→setSleepCountdown()

Modifie le temps avant le prochain sommeil .

```
js      function set_sleepCountdown( newval)
cpp     int set_sleepCountdown( int newval)
m       -(int) setSleepCountdown : (int) newval
pas     integer set_sleepCountdown( newval: LongInt): integer
vb      function set_sleepCountdown( ByVal newval As Integer) As Integer
cs      int set_sleepCountdown( int newval)
dnp     int set_sleepCountdown( int newval)
java    int set_sleepCountdown( int newval)
uwp     async Task<int> set_sleepCountdown( int newval)
py      set_sleepCountdown( newval)
php     function set_sleepCountdown( $newval)
es      async set_sleepCountdown( newval)
cmd     YWakeUpMonitor target set_sleepCountdown newval
```

Paramètres :

newval un entier représentant le temps avant le prochain sommeil

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→set_userData()
wakeupmonitor→setUserData()

YWakeUpMonitor

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js    function set_userData( data)
cpp   void set_userData( void * data)
m     -(void) setUserData : (id) data
pas   set_userData( data: Tobject)
vb    procedure set_userData( ByVal data As Object)
cs    void set_userData( object data)
java  void set_userData( Object data)
py    set_userData( data)
php   function set_userData( $data)
es    async set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Paramètres :

data objet quelconque à mémoriser

wakeupmonitor→sleep() YWakeUpMonitor

Déclenche une mise en sommeil jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

```
js    function sleep( secBeforeSleep)
cpp   int sleep( int secBeforeSleep)
m     -(int) sleep : (int) secBeforeSleep
pas   LongInt sleep( secBeforeSleep: LongInt): LongInt
vb    function sleep( ) As Integer
cs    int sleep( int secBeforeSleep)
dnp   int sleep( int secBeforeSleep)
java  int sleep( int secBeforeSleep)
uwp   async Task<int> sleep( int secBeforeSleep)
py    sleep( secBeforeSleep)
php   function sleep( $secBeforeSleep)
es    async sleep( secBeforeSleep)
cmd   YWakeUpMonitor target sleep secBeforeSleep
```

Paramètres :

secBeforeSleep nombre de seconde avant la mise en sommeil

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→sleepFor() YWakeUpMonitor

Déclenche une mise en sommeil pour un temps donné ou jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

```
js    function sleepFor( secUntilWakeUp, secBeforeSleep)
cpp   int sleepFor( int secUntilWakeUp, int secBeforeSleep)
m     -(int) sleepFor : (int) secUntilWakeUp
                      : (int) secBeforeSleep
pas   LongInt sleepFor( secUntilWakeUp: LongInt,
                        secBeforeSleep: LongInt): LongInt
vb    function sleepFor( ) As Integer
cs    int sleepFor( int secUntilWakeUp, int secBeforeSleep)
dnp   int sleepFor( int secUntilWakeUp, int secBeforeSleep)
java  int sleepFor( int secUntilWakeUp, int secBeforeSleep)
uwp   async Task<int> sleepFor( int secUntilWakeUp, int secBeforeSleep)
py    sleepFor( secUntilWakeUp, secBeforeSleep)
php   function sleepFor( $secUntilWakeUp, $secBeforeSleep)
es    async sleepFor( secUntilWakeUp, secBeforeSleep)
cmd   YWakeUpMonitor target sleepFor secUntilWakeUp secBeforeSleep
```

Le compte à rebours avant la mise en sommeil peut être annulé grâce à resetSleepCountDown.

Paramètres :

secUntilWakeUp nombre de secondes avant le prochain réveil

secBeforeSleep nombre de secondes avant la mise en sommeil

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→sleepUntil() YWakeUpMonitor

Déclenche une mise en sommeil jusqu'à une date donnée ou jusqu'à la prochaine condition de réveil, l'heure du RTC du module doit impérativement avoir été réglée au préalable.

```
js    function sleepUntil( wakeUpTime, secBeforeSleep)
cpp   int sleepUntil( int wakeUpTime, int secBeforeSleep)
m     -(int) sleepUntil : (int) wakeUpTime : (int) secBeforeSleep
pas   LongInt sleepUntil( wakeUpTime: LongInt,
                          secBeforeSleep: LongInt): LongInt
vb    function sleepUntil( ) As Integer
cs    int sleepUntil( int wakeUpTime, int secBeforeSleep)
dnp   int sleepUntil( int wakeUpTime, int secBeforeSleep)
java  int sleepUntil( int wakeUpTime, int secBeforeSleep)
uwp   async Task<int> sleepUntil( int wakeUpTime, int secBeforeSleep)
py    sleepUntil( wakeUpTime, secBeforeSleep)
php   function sleepUntil( $wakeUpTime, $secBeforeSleep)
es    async sleepUntil( wakeUpTime, secBeforeSleep)
cmd   YWakeUpMonitor target sleepUntil wakeUpTime secBeforeSleep
```

Le compte à rebours avant la mise en sommeil peut être annulé grâce à resetSleepCountDown.

Paramètres :

wakeUpTime date/heure du réveil (format UNIX)

secBeforeSleep nombre de secondes avant la mise en sommeil

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→unmuteValueCallbacks() YWakeUpMonitor

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

```
js      function unmuteValueCallbacks( )
cpp     int unmuteValueCallbacks( )
m       -(int) unmuteValueCallbacks
pas     LongInt unmuteValueCallbacks( ): LongInt
vb      function unmuteValueCallbacks( ) As Integer
cs      int unmuteValueCallbacks( )
dnp     int unmuteValueCallbacks( )
java    int unmuteValueCallbacks( )
uwp     async Task<int> unmuteValueCallbacks( )
py      unmuteValueCallbacks( )
php     function unmuteValueCallbacks( )
es      async unmuteValueCallbacks( )
cmd     YWakeUpMonitor target unmuteValueCallbacks
```

Cette fonction annule un précédent appel à `muteValueCallbacks()`. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupmonitor→wait_async() YWakeUpMonitor

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js  function wait_async( callback, context)
es  wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

Paramètres :

callback fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout.

wakeupmonitor→wakeUp() YWakeUpMonitor

Force un réveil.

```
js    function wakeUp( )
cpp   int wakeUp( )
m     -(int) wakeUp
pas   LongInt wakeUp( ): LongInt
vb    function wakeUp( ) As Integer
cs    int wakeUp( )
dnp   int wakeUp( )
java  int wakeUp( )
uwp   async Task<int> wakeUp( )
py    wakeUp( )
php   function wakeUp( )
es    async wakeUp( )
cmd   YWakeUpMonitor target wakeUp
```

8.7. La classe YWakeUpSchedule

Interface pour intéragir avec les réveils agendés, disponibles par exemple dans le YoctoHub-GSM-3G-EU, le YoctoHub-GSM-3G-NA, le YoctoHub-Wireless-SR et le YoctoHub-Wireless-g

La classe `YWakeUpSchedule` implémente une condition de réveil. Le réveil est spécifié par un ensemble de mois et/ou jours et/ou heures et/ou minutes où il doit se produire.

Pour utiliser les fonctions décrites ici, vous devez inclure:

es	in HTML: <script src="../../lib/yocto_wakeupschedule.js"></script> in node.js: require('yoctolib-es2017/yocto_wakeupschedule.js');
js	<script type='text/javascript' src='yocto_wakeupschedule.js'></script>
cpp	#include "yocto_wakeupschedule.h"
m	#import "yocto_wakeupschedule.h"
pas	uses yocto_wakeupschedule;
vb	yocto_wakeupschedule.vb
cs	yocto_wakeupschedule.cs
dnp	import YoctoProxyAPI.YWakeUpScheduleProxy
java	import com.yoctopuce.YoctoAPI.YWakeUpSchedule;
uwp	import com.yoctopuce.YoctoAPI.YWakeUpSchedule;
py	from yocto_wakeupschedule import *
php	require_once('yocto_wakeupschedule.php');
vi	YWakeUpSchedule.vi

Fonction globales

YWakeUpSchedule.FindWakeUpSchedule(**func**)

Permet de retrouver un réveil agendé d'après un identifiant donné.

YWakeUpSchedule.FindWakeUpScheduleInContext(**yctx**, **func**)

Permet de retrouver un réveil agendé d'après un identifiant donné dans un Context YAPI.

YWakeUpSchedule.FirstWakeUpSchedule()

Commence l'énumération des réveils agendés accessibles par la librairie.

YWakeUpSchedule.FirstWakeUpScheduleInContext(**yctx**)

Commence l'énumération des réveils agendés accessibles par la librairie.

YWakeUpSchedule.GetSimilarFunctions()

Enumère toutes les fonctions de type WakeUpSchedule disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).

Propriétés des objets `YWakeUpScheduleProxy`

wakeupschedule→AdvertisedValue *[lecture seule]*

Courte chaîne de caractères représentant l'état courant de la fonction.

wakeupschedule→FriendlyName *[lecture seule]*

Identifiant global de la fonction au format `NOM_MODULE.NOM_FONCTION`.

wakeupschedule→FunctionId *[lecture seule]*

Identifiant matériel du réveil agendé, sans référence au module.

wakeupschedule→HardwareId *[lecture seule]*

Identifiant matériel unique de la fonction au format `SERIAL.FUNCTIONID`.

wakeupschedule→Hours *[modifiable]*

S heures où le réveil est actif..

wakeupschedule→IsOnline *[lecture seule]*

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

wakeupschedule→LogicalName *[modifiable]*

Nom logique de la fonction.

wakeupschedule→MinutesA *[modifiable]*

S minutes de l'interval 00-29 de chaque heures où le réveil est actif.

wakeupschedule→MinutesB *[modifiable]*

S minutes de l'interval 30-59 de chaque heure où le réveil est actif.

wakeupschedule→MonthDays *[modifiable]*

S jours du mois où le réveil est actif.

wakeupschedule→Months *[modifiable]*

S mois où le réveil est actif.

wakeupschedule→NextOccurence *[lecture seule]*

Date/heure de la prochaine occurence de réveil.

wakeupschedule→SerialNumber *[lecture seule]*

Numéro de série du module, préprogrammé en usine.

wakeupschedule→WeekDays *[modifiable]*

S jours de la semaine où le réveil est actif.

Méthodes des objets YWakeUpSchedule

wakeupschedule→clearCache()

Invalide le cache.

wakeupschedule→describe()

Retourne un court texte décrivant de manière non-ambigüe l'instance du réveil agendé au format TYPE(NAME)=SERIAL.FUNCTIONID.

wakeupschedule→get_advertisedValue()

Retourne la valeur courante du réveil agendé (pas plus de 6 caractères).

wakeupschedule→get_errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.

wakeupschedule→get_errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.

wakeupschedule→get_friendlyName()

Retourne un identifiant global du réveil agendé au format NOM_MODULE.NOM_FONCTION.

wakeupschedule→get_functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

wakeupschedule→get_functionId()

Retourne l'identifiant matériel du réveil agendé, sans référence au module.

wakeupschedule→get_hardwareId()

Retourne l'identifiant matériel unique du réveil agendé au format SERIAL.FUNCTIONID.

wakeupschedule→get_hours()

Retourne les heures où le réveil est actif..

wakeupschedule→get_logicalName()

Retourne le nom logique du réveil agendé.

wakeupschedule→get_minutes()

Retourne toutes les minutes de chaque heure où le réveil est actif.

wakeupschedule→get_minutesA()

Retourne les minutes de l'interval 00-29 de chaque heures où le réveil est actif.

wakeupschedule→get_minutesB()

Retourne les minutes de l'interval 30-59 de chaque heure où le réveil est actif.

wakeupschedule→get_module()

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

wakeupschedule→get_module_async(**callback**, **context**)

Retourne l'objet `YModule` correspondant au module Yoctopuce qui héberge la fonction.

wakeupschedule→get_monthDays()

Retourne les jours du mois où le réveil est actif.

wakeupschedule→get_months()

Retourne les mois où le réveil est actif.

wakeupschedule→get_nextOccurence()

Retourne la date/heure de la prochaine occurence de réveil.

wakeupschedule→get_serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

wakeupschedule→get_userData()

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

wakeupschedule→get_weekDays()

Retourne les jours de la semaine où le réveil est actif.

wakeupschedule→isOnline()

Vérifie si le module hébergeant le réveil agendé est joignable, sans déclencher d'erreur.

wakeupschedule→isOnline_async(**callback**, **context**)

Vérifie si le module hébergeant le réveil agendé est joignable, sans déclencher d'erreur.

wakeupschedule→isReadOnly()

Test si la fonction est en lecture seule.

wakeupschedule→load(**msValidity**)

Met en cache les valeurs courantes du réveil agendé, avec une durée de validité spécifiée.

wakeupschedule→loadAttribute(**attrName**)

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

wakeupschedule→load_async(**msValidity**, **callback**, **context**)

Met en cache les valeurs courantes du réveil agendé, avec une durée de validité spécifiée.

wakeupschedule→muteValueCallbacks()

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

wakeupschedule→nextWakeUpSchedule()

Continue l'énumération des réveils agendés commencée à l'aide de `yFirstWakeUpSchedule()`
Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les réveils agendés sont retournés.

wakeupschedule→registerValueCallback(**callback**)

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

wakeupschedule→set_hours(**newval**)

Modifie les heures où un réveil doit avoir lieu.

wakeupschedule→set_logicalName(**newval**)

Modifie le nom logique du réveil agendé.

wakeupschedule→set_minutes(**bitmap**)

Modifie toutes les minutes où un réveil doit avoir lieu

wakeupschedule→set_minutesA(**newval**)

Modifie les minutes de l'interval 00-29 où un réveil doit avoir lieu.

wakeupschedule→set_minutesB(newval)

Modifie les minutes de l'interval 30-59 où un réveil doit avoir lieu.

wakeupschedule→set_monthDays(newval)

Modifie les jours du mois où un réveil doit avoir lieu.

wakeupschedule→set_months(newval)

Modifie les mois où un réveil doit avoir lieu.

wakeupschedule→set_userData(data)

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

wakeupschedule→set_weekDays(newval)

Modifie les jours de la semaine où un réveil doit avoir lieu.

wakeupschedule→unmuteValueCallbacks()

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

wakeupschedule→wait_async(callback, context)

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

YWakeUpSchedule.FindWakeUpSchedule() YWakeUpSchedule
YWakeUpSchedule.FindWakeUpSchedule()

Permet de retrouver un réveil agendé d'après un identifiant donné.

```
js      function yFindWakeUpSchedule( func)
cpp     YWakeUpSchedule* yFindWakeUpSchedule( string func)
m       +(YWakeUpSchedule*) FindWakeUpSchedule : (NSString*) func
pas     TYWakeUpSchedule yFindWakeUpSchedule( func: string): TYWakeUpSchedule
vb      function yFindWakeUpSchedule( ByVal func As String) As YWakeUpSchedule
cs      static YWakeUpSchedule FindWakeUpSchedule( string func)
dnp     static YWakeUpScheduleProxy FindWakeUpSchedule( string func)
java    static YWakeUpSchedule FindWakeUpSchedule( String func)
uwp     static YWakeUpSchedule FindWakeUpSchedule( string func)
py      FindWakeUpSchedule( func)
php     function yFindWakeUpSchedule( $func)
es      static FindWakeUpSchedule( func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le réveil agendé soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YWakeUpSchedule.isOnline()` pour tester si le réveil agendé est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Si un appel à la méthode is_online() de cet objet renvoie FAUX alors que vous êtes sûr que le module correspondant est bien branché, vérifiez que vous n'avez pas oublié d'appeler registerHub() à l'initialisation de de l'application.

Paramètres :

func une chaîne de caractères qui référence le réveil agendé sans ambiguïté, par exemple `YHUBGSM3.wakeUpSchedule1`.

Retourne :

un objet de classe `YWakeUpSchedule` qui permet ensuite de contrôler le réveil agendé.

YWakeUpSchedule.FindWakeUpScheduleInContext() YWakeUpSchedule YWakeUpSchedule.FindWakeUpScheduleInContext()

Permet de retrouver un réveil agendé d'après un identifiant donné dans un Context YAPI.

```
java  static YWakeUpSchedule FindWakeUpScheduleInContext( YAPIContext yctx,
                                                          String func)
uwp   static YWakeUpSchedule FindWakeUpScheduleInContext( YAPIContext yctx,
                                                          string func)
es    static FindWakeUpScheduleInContext( yctx, func)
```

L'identifiant peut être spécifié sous plusieurs formes:

- NomLogiqueFonction
- NoSerieModule.IdentifiantFonction
- NoSerieModule.NomLogiqueFonction
- NomLogiqueModule.IdentifiantMatériel
- NomLogiqueModule.NomLogiqueFonction

Cette fonction n'exige pas que le réveil agendé soit en ligne au moment ou elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YWakeUpSchedule.isOnline()` pour tester si le réveil agendé est utilisable à un moment donné. En cas d'ambiguïté lorsqu'on fait une recherche par nom logique, aucune erreur ne sera notifiée: la première instance trouvée sera renvoyée. La recherche se fait d'abord par nom matériel, puis par nom logique.

Paramètres :

yctx un contexte YAPI

func une chaîne de caractères qui référence le réveil agendé sans ambiguïté, par exemple `YHUBGSM3.wakeUpSchedule1`.

Retourne :

un objet de classe `YWakeUpSchedule` qui permet ensuite de contrôler le réveil agendé.

YWakeUpSchedule.FirstWakeUpSchedule() YWakeUpSchedule YWakeUpSchedule.FirstWakeUpSchedule()

Commence l'énumération des réveils agendés accessibles par la librairie.

js	function **yFirstWakeUpSchedule**( )
cpp	YWakeUpSchedule * **yFirstWakeUpSchedule**( )
m	+(YWakeUpSchedule*) **FirstWakeUpSchedule**
pas	TYWakeUpSchedule **yFirstWakeUpSchedule**( ): TYWakeUpSchedule
vb	function **yFirstWakeUpSchedule**( ) As YWakeUpSchedule
cs	static YWakeUpSchedule **FirstWakeUpSchedule**( )
java	static YWakeUpSchedule **FirstWakeUpSchedule**( )
uwp	static YWakeUpSchedule **FirstWakeUpSchedule**( )
py	**FirstWakeUpSchedule**( )
php	function **yFirstWakeUpSchedule**( )
es	static **FirstWakeUpSchedule**( )

Utiliser la fonction `YWakeUpSchedule.nextWakeUpSchedule()` pour itérer sur les autres réveils agendés.

Retourne :

un pointeur sur un objet `YWakeUpSchedule`, correspondant au premier réveil agendé accessible en ligne, ou `null` si il n'y a pas de réveils agendés disponibles.

YWakeUpSchedule.FirstWakeUpScheduleInContext() YWakeUpSchedule YWakeUpSchedule.FirstWakeUpScheduleInContext()

Commence l'énumération des réveils agendés accessibles par la librairie.

java static YWakeUpSchedule **FirstWakeUpScheduleInContext**(YAPIContext **yctx**)
uwp static YWakeUpSchedule **FirstWakeUpScheduleInContext**(YAPIContext **yctx**)
es static **FirstWakeUpScheduleInContext**(**yctx**)

Utiliser la fonction `YWakeUpSchedule.nextWakeUpSchedule()` pour itérer sur les autres réveils agendés.

Paramètres :

yctx un contexte YAPI.

Retourne :

un pointeur sur un objet `YWakeUpSchedule`, correspondant au premier réveil agendé accessible en ligne, ou `null` si il n'y a pas de réveils agendés disponibles.

YWakeUpSchedule.GetSimilarFunctions() YWakeUpSchedule YWakeUpSchedule.GetSimilarFunctions()

Enumère toutes les fonctions de type WakeUpSchedule disponibles sur les modules actuellement joignables par la librairie, et retourne leurs identifiants matériels uniques (hardwareId).

dnp static new string[] **GetSimilarFunctions**( )

Chaque chaîne retournée peut être passée en argument à la méthode `YWakeUpSchedule.FindWakeUpSchedule` pour obtenir une objet permettant d'intéragir avec le module correspondant.

Retourne :

un tableau de chaînes de caractères, contenant les identifiants matériels de chaque fonction disponible trouvée.

wakeupschedule→AdvertisedValue — YWakeUpSchedule

Courte chaîne de caractères représentant l'état courant de la fonction.

dnp string **AdvertisedValue**

wakeupschedule→FriendlyName — YWakeUpSchedule

Identifiant global de la fonction au format NOM_MODULE.NOM_FONCTION.

dnp string **FriendlyName**

Le chaîne retournée utilise soit les noms logiques du module et de la fonction si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel de la fonction (par exemple: MyCustomName.relay1)

wakeupschedule→FunctionId — YWakeUpSchedule

Identifiant matériel du réveil agendé, sans référence au module.

dnp string **FunctionId**

Par example `relay1`.

wakeupschedule→HardwareId — YWakeUpSchedule

Identifiant matériel unique de la fonction au format SERIAL.FUNCTIONID.

dnp string **HardwareId**

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de la fonction (par example RELAYLO1-123456.relay1).

wakeupschedule→Hours — YWakeUpSchedule

S heures où le réveil est actif..

dnp int **Hours**

Modifiable. Modifie les heures où un réveil doit avoir lieu. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

wakeupschedule→IsOnline — YWakeUpSchedule

Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.

dnp bool **IsOnline**

Si les valeurs des attributs en cache de la fonction sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

wakeupschedule→LogicalName — YWakeUpSchedule

Nom logique de la fonction.

dnp string **LogicalName**

Modifiable. Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

wakeupschedule→MinutesA — YWakeUpSchedule

S minutes de l'interval 00-29 de chaque heures où le réveil est actif.

dnp int **MinutesA**

Modifiable. Modifie les minutes de l'interval 00-29 où un réveil doit avoir lieu. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

wakeupschedule→MinutesB — YWakeUpSchedule

S minutes de l'interval 30-59 de chaque heure où le réveil est actif.

dnp int **MinutesB**

Modifiable. Modifie les minutes de l'interval 30-59 où un réveil doit avoir lieu. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

wakeupschedule→MonthDays — YWakeUpSchedule

S jours du mois où le réveil est actif.

dnp int **MonthDays**

Modifiable. Modifie les jours du mois où un réveil doit avoir lieu. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

wakeupschedule→Months YWakeUpSchedule

S mois où le réveil est actif.

dnp int **Months**

Modifiable. Modifie les mois où un réveil doit avoir lieu. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

wakeupschedule→NextOccurence YWakeUpSchedule

Date/heure de la prochaine occurence de réveil.

dnp long **NextOccurence**

wakeupschedule→SerialNumber — YWakeUpSchedule

Numéro de série du module, préprogrammé en usine.

dnp string **SerialNumber**

wakeupschedule→WeekDays YWakeUpSchedule

S jours de la semaine où le réveil est actif.

dnp int **WeekDays**

Modifiable. Modifie les jours de la semaine où un réveil doit avoir lieu. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

wakeupschedule→clearCache() YWakeUpSchedule

Invalide le cache.

js	function **clearCache**( )
cpp	void **clearCache**( )
m	-(void) **clearCache**
pas	**clearCache**( )
vb	procedure **clearCache**( )
cs	void **clearCache**( )
java	void **clearCache**( )
py	**clearCache**( )
php	function **clearCache**( )
es	async **clearCache**( )

Invalide le cache des valeurs courantes du réveil agendé. Force le prochain appel à une méthode get_xxx() ou loadxxx() pour charger les les données depuis le module.

wakeupschedule→describe() YWakeUpSchedule

Retourne un court texte décrivant de manière non-ambigüe l'instance du réveil agendé au format TYPE(NAME)=SERIAL.FUNCTIONID.

js	function **describe**( )
cpp	string **describe**( )
m	-(NSString*) **describe**
pas	string **describe**( ): string
vb	function **describe**( ) As String
cs	string **describe**( )
java	String **describe**( )
py	**describe**( )
php	function **describe**( )
es	async **describe**( )

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilsé lors du premier accès a la fonction, SERIAL correspond au numéro de série du module si le module est connecté, ou "unresolved" sinon, et FUNCTIONID correspond à l'identifiant matériel de la fonction si le module est connecté. Par exemple, La methode va retourner Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1 si le module est déjà connecté ou Relay(BadCustomeName.relay1)=unresolved si le module n'est pas déjà connecté. Cette methode ne declenche aucune transaction USB ou TCP et peut donc être utilisé dans un debuggeur.

Retourne :
une chaîne de caractères décrivant le réveil agendé (ex: Relay(MyCustomName.relay1)=RELAYLO1-123456.relay1)

wakeupschedule→get_advertisedValue() YWakeUpSchedule
wakeupschedule→advertisedValue()

Retourne la valeur courante du réveil agendé (pas plus de 6 caractères).

```
js      function get_advertisedValue( )
cpp     string get_advertisedValue( )
m       -(NSString*) advertisedValue
pas     string get_advertisedValue( ): string
vb      function get_advertisedValue( ) As String
cs      string get_advertisedValue( )
dnp     string get_advertisedValue( )
java    String get_advertisedValue( )
uwp     async Task<string> get_advertisedValue( )
py      get_advertisedValue( )
php     function get_advertisedValue( )
es      async get_advertisedValue( )
cmd     YWakeUpSchedule target get_advertisedValue
```

Retourne :
une chaîne de caractères représentant la valeur courante du réveil agendé (pas plus de 6 caractères).

En cas d'erreur, déclenche une exception ou retourne `Y_ADVERTISEDVALUE_INVALID`.

wakeupschedule→get_errorMessage() YWakeUpSchedule
wakeupschedule→errorMessage()

Retourne le message correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.

js	function **get_errorMessage**( )
cpp	string **get_errorMessage**( )
m	-(NSString*) errorMessage
pas	string **get_errorMessage**( ): string
vb	function **get_errorMessage**( ) As String
cs	string **get_errorMessage**( )
java	String **get_errorMessage**( )
py	**get_errorMessage**( )
php	function **get_errorMessage**( )
es	**get_errorMessage**( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
une chaîne de caractères correspondant au message de la dernière erreur qui s'est produit lors de l'utilisation du réveil agendé.

wakeupschedule→get_errorType() YWakeUpSchedule
wakeupschedule→errorType()

Retourne le code d'erreur correspondant à la dernière erreur survenue lors de l'utilisation du réveil agendé.

js	function **get_errorType**( )
cpp	YRETCODE **get_errorType**( )
m	-(YRETCODE) errorType
pas	YRETCODE **get_errorType**( ): YRETCODE
vb	function **get_errorType**( ) As YRETCODE
cs	YRETCODE **get_errorType**( )
java	int **get_errorType**( )
py	**get_errorType**( )
php	function **get_errorType**( )
es	**get_errorType**( )

Cette méthode est principalement utile lorsque la librairie Yoctopuce est utilisée en désactivant la gestion des exceptions.

Retourne :
un nombre correspondant au code de la dernière erreur qui s'est produit lors de l'utilisation du réveil agendé.

wakeupschedule→get_friendlyName() YWakeUpSchedule
wakeupschedule→friendlyName()

Retourne un identifiant global du réveil agendé au format NOM_MODULE.NOM_FONCTION.

js	function **get_friendlyName**( )
cpp	string **get_friendlyName**( )
m	-(NSString*) friendlyName
cs	string **get_friendlyName**( )
dnp	string **get_friendlyName**( )
java	String **get_friendlyName**( )
py	**get_friendlyName**( )
php	function **get_friendlyName**( )
es	async **get_friendlyName**( )

Le chaîne retournée utilise soit les noms logiques du module et du réveil agendé si ils sont définis, soit respectivement le numéro de série du module et l'identifant matériel du réveil agendé (par exemple: MyCustomName.relay1)

Retourne :
une chaîne de caractères identifiant le réveil agendé en utilisant les noms logiques (ex: MyCustomName.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_FRIENDLYNAME_INVALID.

wakeupschedule→get_functionDescriptor() YWakeUpSchedule
wakeupschedule→functionDescriptor()

Retourne un identifiant unique de type YFUN_DESCR correspondant à la fonction.

js	function **get_functionDescriptor**( )
cpp	YFUN_DESCR **get_functionDescriptor**( )
m	-(YFUN_DESCR) functionDescriptor
pas	YFUN_DESCR **get_functionDescriptor**( ): YFUN_DESCR
vb	function **get_functionDescriptor**( ) As YFUN_DESCR
cs	YFUN_DESCR **get_functionDescriptor**( )
java	String **get_functionDescriptor**( )
py	**get_functionDescriptor**( )
php	function **get_functionDescriptor**( )
es	async **get_functionDescriptor**( )

Cet identifiant peut être utilisé pour tester si deux instance de YFunction référencent physiquement la même fonction sur le même module.

Retourne :
un identifiant de type YFUN_DESCR.

Si la fonction n'a jamais été contactée, la valeur retournée sera Y_FUNCTIONDESCRIPTOR_INVALID

wakeupschedule→get_functionId()
wakeupschedule→functionId()

YWakeUpSchedule

Retourne l'identifiant matériel du réveil agendé, sans référence au module.

js	function **get_functionId**( )
cpp	string **get_functionId**( )
m	-(NSString*) functionId
vb	function **get_functionId**( ) As String
cs	string **get_functionId**( )
dnp	string **get_functionId**( )
java	String **get_functionId**( )
py	**get_functionId**( )
php	function **get_functionId**( )
es	async **get_functionId**( )

Par example `relay1`.

Retourne :
une chaîne de caractères identifiant le réveil agendé (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `Y_FUNCTIONID_INVALID`.

wakeupschedule→get_hardwareId()
wakeupschedule→hardwareId()

YWakeUpSchedule

Retourne l'identifiant matériel unique du réveil agendé au format SERIAL.FUNCTIONID.

```
js      function get_hardwareId( )
cpp     string get_hardwareId( )
m       -(NSString*) hardwareId
vb      function get_hardwareId( ) As String
cs      string get_hardwareId( )
dnp     string get_hardwareId( )
java    String get_hardwareId( )
py      get_hardwareId( )
php     function get_hardwareId( )
es      async get_hardwareId( )
```

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du réveil agendé (par example RELAYLO1-123456.relay1).

Retourne :
une chaîne de caractères identifiant le réveil agendé (ex: RELAYLO1-123456.relay1)

En cas d'erreur, déclenche une exception ou retourne Y_HARDWAREID_INVALID.

wakeupschedule→get_hours()
wakeupschedule→hours()

YWakeUpSchedule

Retourne les heures où le réveil est actif..

```
js    function get_hours( )
cpp   int get_hours( )
m     -(int) hours
pas   LongInt get_hours( ): LongInt
vb    function get_hours( ) As Integer
cs    int get_hours( )
dnp   int get_hours( )
java  int get_hours( )
uwp   async Task<int> get_hours( )
py    get_hours( )
php   function get_hours( )
es    async get_hours( )
cmd   YWakeUpSchedule target get_hours
```

Retourne :

un entier représentant les heures où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne `Y_HOURS_INVALID`.

wakeupschedule→get_logicalName()
wakeupschedule→logicalName()

YWakeUpSchedule

Retourne le nom logique du réveil agendé.

```
js      function get_logicalName( )
cpp     string get_logicalName( )
m       -(NSString*) logicalName
pas     string get_logicalName( ): string
vb      function get_logicalName( ) As String
cs      string get_logicalName( )
dnp     string get_logicalName( )
java    String get_logicalName( )
uwp     async Task<string> get_logicalName( )
py      get_logicalName( )
php     function get_logicalName( )
es      async get_logicalName( )
cmd     YWakeUpSchedule target get_logicalName
```

Retourne :
une chaîne de caractères représentant le nom logique du réveil agendé.

En cas d'erreur, déclenche une exception ou retourne `Y_LOGICALNAME_INVALID`.

wakeupschedule→get_minutes() YWakeUpSchedule
wakeupschedule→minutes()

Retourne toutes les minutes de chaque heure où le réveil est actif.

```
js      function get_minutes( )
cpp     s64 get_minutes( )
m       -(s64) minutes
pas     int64 get_minutes( ): int64
vb      function get_minutes( ) As Long
cs      long get_minutes( )
dnp     long get_minutes( )
java    long get_minutes( )
uwp     async Task<long> get_minutes( )
py      get_minutes( )
php     function get_minutes( )
es      async get_minutes( )
cmd     YWakeUpSchedule target get_minutes
```

wakeupschedule→get_minutesA() YWakeUpSchedule
wakeupschedule→minutesA()

Retourne les minutes de l'interval 00-29 de chaque heures où le réveil est actif.

```
js    function get_minutesA( )
cpp   int get_minutesA( )
m     -(int) minutesA
pas   LongInt get_minutesA( ): LongInt
vb    function get_minutesA( ) As Integer
cs    int get_minutesA( )
dnp   int get_minutesA( )
java  int get_minutesA( )
uwp   async Task<int> get_minutesA( )
py    get_minutesA( )
php   function get_minutesA( )
es    async get_minutesA( )
cmd   YWakeUpSchedule target get_minutesA
```

Retourne :
un entier représentant les minutes de l'interval 00-29 de chaque heures où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne `Y_MINUTESA_INVALID`.

wakeupschedule→get_minutesB() YWakeUpSchedule
wakeupschedule→minutesB()

Retourne les minutes de l'interval 30-59 de chaque heure où le réveil est actif.

js	function **get_minutesB**( )
cpp	int **get_minutesB**( )
m	-(int) minutesB
pas	LongInt **get_minutesB**( ): LongInt
vb	function **get_minutesB**( ) As Integer
cs	int **get_minutesB**( )
dnp	int **get_minutesB**( )
java	int **get_minutesB**( )
uwp	async Task<int> **get_minutesB**( )
py	**get_minutesB**( )
php	function **get_minutesB**( )
es	async **get_minutesB**( )
cmd	YWakeUpSchedule **target get_minutesB**

Retourne :
un entier représentant les minutes de l'interval 30-59 de chaque heure où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne `Y_MINUTESB_INVALID`.

wakeupschedule→get_module()
wakeupschedule→module()

YWakeUpSchedule

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

js	function **get_module**( )
cpp	YModule * **get_module**( )
m	-(YModule*) module
pas	TYModule **get_module**( ): TYModule
vb	function **get_module**( ) As YModule
cs	YModule **get_module**( )
dnp	YModuleProxy **get_module**( )
java	YModule **get_module**( )
py	**get_module**( )
php	function **get_module**( )
es	async **get_module**( )

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Retourne :
une instance de YModule

wakeupschedule→get_module_async() YWakeUpSchedule
wakeupschedule→module_async()

Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.

```
js  function get_module_async( callback, context)
```

Si la fonction ne peut être trouvée sur aucun module, l'instance de YModule retournée ne sera pas joignable.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la VM Javascript de Firefox, qui n'implémente pas le passage de contrôle entre threads durant les appels d'entrée/sortie bloquants.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et l'instance demandée de YModule

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wakeupschedule→get_monthDays()
wakeupschedule→monthDays()

YWakeUpSchedule

Retourne les jours du mois où le réveil est actif.

```
js      function get_monthDays( )
cpp     int get_monthDays( )
m       -(int) monthDays
pas     LongInt get_monthDays( ): LongInt
vb      function get_monthDays( ) As Integer
cs      int get_monthDays( )
dnp     int get_monthDays( )
java    int get_monthDays( )
uwp     async Task<int> get_monthDays( )
py      get_monthDays( )
php     function get_monthDays( )
es      async get_monthDays( )
cmd     YWakeUpSchedule target get_monthDays
```

Retourne :
un entier représentant les jours du mois où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne `Y_MONTHDAYS_INVALID`.

wakeupschedule→get_months()
wakeupschedule→months()

YWakeUpSchedule

Retourne les mois où le réveil est actif.

```
js      function get_months( )
cpp     int get_months( )
m       -(int) months
pas     LongInt get_months( ): LongInt
vb      function get_months( ) As Integer
cs      int get_months( )
dnp     int get_months( )
java    int get_months( )
uwp     async Task<int> get_months( )
py      get_months( )
php     function get_months( )
es      async get_months( )
cmd     YWakeUpSchedule target get_months
```

Retourne :
un entier représentant les mois où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne `Y_MONTHS_INVALID`.

wakeupschedule→get_nextOccurence()
wakeupschedule→nextOccurence()

YWakeUpSchedule

Retourne la date/heure de la prochaine occurence de réveil.

```
js      function get_nextOccurence( )
cpp     s64 get_nextOccurence( )
m       -(s64) nextOccurence
pas     int64 get_nextOccurence( ): int64
vb      function get_nextOccurence( ) As Long
cs      long get_nextOccurence( )
dnp     long get_nextOccurence( )
java    long get_nextOccurence( )
uwp     async Task<long> get_nextOccurence( )
py      get_nextOccurence( )
php     function get_nextOccurence( )
es      async get_nextOccurence( )
cmd     YWakeUpSchedule target get_nextOccurence
```

Retourne :
un entier représentant la date/heure de la prochaine occurence de réveil

En cas d'erreur, déclenche une exception ou retourne `Y_NEXTOCCURENCE_INVALID`.

wakeupschedule→get_serialNumber() YWakeUpSchedule
wakeupschedule→serialNumber()

Retourne le numéro de série du module, préprogrammé en usine.

```
js      function get_serialNumber( )
cpp     string get_serialNumber( )
m       -(NSString*) serialNumber
pas     string get_serialNumber( ): string
vb      function get_serialNumber( ) As String
cs      string get_serialNumber( )
dnp     string get_serialNumber( )
java    String get_serialNumber( )
uwp     async Task<string> get_serialNumber( )
py      get_serialNumber( )
php     function get_serialNumber( )
es      async get_serialNumber( )
cmd     YWakeUpSchedule target get_serialNumber
```

Retourne :

: une chaîne de caractères représentant le numéro de série du module, préprogrammé en usine.

En cas d'erreur, déclenche une exception ou retourne YModule.SERIALNUMBER_INVALID.

wakeupschedule→get_userData()
wakeupschedule→userData()

YWakeUpSchedule

Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode `set_userData`.

```
js      function get_userData( )
cpp     void * get_userData( )
m       -(id) userData
pas     Tobject get_userData( ): Tobject
vb      function get_userData( ) As Object
cs      object get_userData( )
java    Object get_userData( )
py      get_userData( )
php     function get_userData( )
es      async get_userData( )
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Retourne :
l'objet stocké précédemment par l'appelant.

wakeupschedule→get_weekDays()
wakeupschedule→weekDays()

YWakeUpSchedule

Retourne les jours de la semaine où le réveil est actif.

```
js      function get_weekDays( )
cpp     int get_weekDays( )
m       -(int) weekDays
pas     LongInt get_weekDays( ): LongInt
vb      function get_weekDays( ) As Integer
cs      int get_weekDays( )
dnp     int get_weekDays( )
java    int get_weekDays( )
uwp     async Task<int> get_weekDays( )
py      get_weekDays( )
php     function get_weekDays( )
es      async get_weekDays( )
cmd     YWakeUpSchedule target get_weekDays
```

Retourne :

un entier représentant les jours de la semaine où le réveil est actif

En cas d'erreur, déclenche une exception ou retourne `Y_WEEKDAYS_INVALID`.

wakeupschedule→isOnline() YWakeUpSchedule

Vérifie si le module hébergeant le réveil agendé est joignable, sans déclencher d'erreur.

```
js      function isOnline( )
cpp     bool isOnline( )
m       -(BOOL) isOnline
pas     boolean isOnline( ): boolean
vb      function isOnline( ) As Boolean
cs      bool isOnline( )
dnp     bool isOnline( )
java    boolean isOnline( )
py      isOnline( )
php     function isOnline( )
es      async isOnline( )
```

Si les valeurs des attributs en cache du réveil agendé sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Retourne :
`true` si le réveil agendé est joignable, `false` sinon

wakeupschedule→isOnline_async() YWakeUpSchedule

Vérifie si le module hébergeant le réveil agendé est joignable, sans déclencher d'erreur.

```
js  function isOnline_async( callback, context)
```

Si les valeurs des attributs en cache du réveil agendé sont valides au moment de l'appel, le module est considéré joignable. Cette fonction ne cause en aucun cas d'exception, quelle que soit l'erreur qui pourrait se produire lors de la vérification de joignabilité.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

callback fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le résultat booléen

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wakeupschedule→isReadOnly() YWakeUpSchedule

Test si la fonction est en lecture seule.

cpp	bool **isReadOnly**( )
m	-(bool) **isReadOnly**
pas	boolean **isReadOnly**( ): boolean
vb	function **isReadOnly**( ) As Boolean
cs	bool **isReadOnly**( )
dnp	bool **isReadOnly**( )
java	boolean **isReadOnly**( )
uwp	async Task<bool> **isReadOnly**( )
py	**isReadOnly**( )
php	function **isReadOnly**( )
es	async **isReadOnly**( )
cmd	YWakeUpSchedule **target isReadOnly**

Retourne vrais si la fonction est protégé en ecriture ou que la fontion n'est pas disponible.

Retourne :
`true` si la fonction est protégé en ecriture ou que la fontion n'est pas disponible

wakeupschedule→load() YWakeUpSchedule

Met en cache les valeurs courantes du réveil agendé, avec une durée de validité spécifiée.

js	function **load**(**msValidity**)
cpp	YRETCODE **load**(int **msValidity**)
m	-(YRETCODE) **load** : (u64) **msValidity**
pas	YRETCODE **load**(**msValidity**: u64): YRETCODE
vb	function **load**(ByVal **msValidity** As Long) As YRETCODE
cs	YRETCODE **load**(ulong **msValidity**)
java	int **load**(long **msValidity**)
py	**load**(**msValidity**)
php	function **load**(**$msValidity**)
es	async **load**(**msValidity**)

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Paramètres :

msValidity un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→loadAttribute() YWakeUpSchedule

Retourne la valeur actuelle d'un attribut spécifique de la fonction, sous forme de texte, le plus rapidement possible mais sans passer par le cache.

```
js      function loadAttribute( attrName)
cpp     string loadAttribute( string attrName)
m       -(NSString*) loadAttribute : (NSString*) attrName
pas     string loadAttribute( attrName: string): string
vb      function loadAttribute( ) As String
cs      string loadAttribute( string attrName)
dnp     string loadAttribute( string attrName)
java    String loadAttribute( String attrName)
uwp     async Task<string> loadAttribute( string attrName)
py      loadAttribute( attrName)
php     function loadAttribute( $attrName)
es      async loadAttribute( attrName)
```

Paramètres :

attrName le nom de l'attribut désiré

Retourne :

une chaîne de caractères représentant la valeur actuelle de l'attribut.

En cas d'erreur, déclenche une exception ou retourne un chaîne vide.

wakeupschedule→load_async() YWakeUpSchedule

Met en cache les valeurs courantes du réveil agendé, avec une durée de validité spécifiée.

```
js  function load_async( msValidity, callback, context)
```

Par défaut, lorsqu'on accède à un module, tous les attributs des fonctions du module sont automatiquement mises en cache pour la durée standard (5 ms). Cette méthode peut être utilisée pour marquer occasionellement les données cachées comme valides pour une plus longue période, par exemple dans le but de réduire le trafic réseau.

Cette version asynchrone n'existe qu'en Javascript. Elle utilise une fonction de callback plutôt qu'une simple valeur de retour, pour éviter de bloquer la machine virtuelle Javascript avec une attente active.

Paramètres :

msValidity	un entier correspondant à la durée de validité attribuée aux les paramètres chargés, en millisecondes
callback	fonction de callback qui sera appelée dès que le résultat sera connu. La fonction callback reçoit trois arguments: le contexte fourni par l'appelant, l'objet fonction concerné et le code d'erreur (ou YAPI_SUCCESS)
context	contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout : le résultat sera passé en paramètre à la fonction de callback.

wakeupschedule→muteValueCallbacks() YWakeUpSchedule

Désactive l'envoi de chaque changement de la valeur publiée au hub parent.

js	function **muteValueCallbacks**( )
cpp	int **muteValueCallbacks**( )
m	-(int) **muteValueCallbacks**
pas	LongInt **muteValueCallbacks**( ): LongInt
vb	function **muteValueCallbacks**( ) As Integer
cs	int **muteValueCallbacks**( )
dnp	int **muteValueCallbacks**( )
java	int **muteValueCallbacks**( )
uwp	async Task<int> **muteValueCallbacks**( )
py	**muteValueCallbacks**( )
php	function **muteValueCallbacks**( )
es	async **muteValueCallbacks**( )
cmd	YWakeUpSchedule **target muteValueCallbacks**

Vous pouvez utiliser cette fonction pour économiser la bande passante et le CPU sur les machines de faible puissance, ou pour éviter le déclanchement de callbacks HTTP. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→nextWakeUpSchedule() YWakeUpSchedule

Continue l'énumération des réveils agendés commencée à l'aide de `yFirstWakeUpSchedule()`. Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les réveils agendés sont retournés.

js	function **nextWakeUpSchedule**( )
cpp	YWakeUpSchedule * **nextWakeUpSchedule**( )
m	-(YWakeUpSchedule*) **nextWakeUpSchedule**
pas	TYWakeUpSchedule **nextWakeUpSchedule**( ): TYWakeUpSchedule
vb	function **nextWakeUpSchedule**( ) As YWakeUpSchedule
cs	YWakeUpSchedule **nextWakeUpSchedule**( )
java	YWakeUpSchedule **nextWakeUpSchedule**( )
uwp	YWakeUpSchedule **nextWakeUpSchedule**( )
py	**nextWakeUpSchedule**( )
php	function **nextWakeUpSchedule**( )
es	**nextWakeUpSchedule**( )

Si vous souhaitez retrouver un réveil agendé spécifique, utilisez `WakeUpSchedule.findWakeUpSchedule()` avec un hardwareID ou un nom logique.

Retourne :

un pointeur sur un objet `YWakeUpSchedule` accessible en ligne, ou `null` lorsque l'énumération est terminée.

wakeupschedule→registerValueCallback() YWakeUpSchedule

Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.

js	function **registerValueCallback**(**callback**)
cpp	int **registerValueCallback**(YWakeUpScheduleValueCallback **callback**)
m	-(int) **registerValueCallback** : (YWakeUpScheduleValueCallback) **callback**
pas	LongInt **registerValueCallback**(**callback**: TYWakeUpScheduleValueCallback): LongInt
vb	function **registerValueCallback**( ) As Integer
cs	int **registerValueCallback**(ValueCallback **callback**)
java	int **registerValueCallback**(UpdateCallback **callback**)
uwp	async Task<int> **registerValueCallback**(ValueCallback **callback**)
py	**registerValueCallback**(**callback**)
php	function **registerValueCallback**(**$callback**)
es	async **registerValueCallback**(**callback**)

Ce callback n'est appelé que durant l'exécution de `ySleep` ou `yHandleEvents`. Cela permet à l'appelant de contrôler quand les callback peuvent se produire. Il est important d'appeler l'une de ces deux fonctions périodiquement pour garantir que les callback ne soient pas appelés trop tard. Pour désactiver un callback, il suffit d'appeler cette méthode en lui passant un pointeur nul.

Paramètres :

callback la fonction de callback à rappeler, ou un pointeur nul. La fonction de callback doit accepter deux arguments: l'object fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

wakeupschedule→set_hours()
wakeupschedule→setHours()

YWakeUpSchedule

Modifie les heures où un réveil doit avoir lieu.

```
js    function set_hours( newval)
cpp   int set_hours( int newval)
m     -(int) setHours : (int) newval
pas   integer set_hours( newval: LongInt): integer
vb    function set_hours( ByVal newval As Integer) As Integer
cs    int set_hours( int newval)
dnp   int set_hours( int newval)
java  int set_hours( int newval)
uwp   async Task<int> set_hours( int newval)
py    set_hours( newval)
php   function set_hours( $newval)
es    async set_hours( newval)
cmd   YWakeUpSchedule target set_hours newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant les heures où un réveil doit avoir lieu

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_logicalName()
wakeupschedule→setLogicalName()

YWakeUpSchedule

Modifie le nom logique du réveil agendé.

```
js    function set_logicalName( newval)
cpp   int set_logicalName( string newval)
m     -(int) setLogicalName : (NSString*) newval
pas   integer set_logicalName( newval: string): integer
vb    function set_logicalName( ByVal newval As String) As Integer
cs    int set_logicalName( string newval)
dnp   int set_logicalName( string newval)
java  int set_logicalName( String newval)
uwp   async Task<int> set_logicalName( string newval)
py    set_logicalName( newval)
php   function set_logicalName( $newval)
es    async set_logicalName( newval)
cmd   YWakeUpSchedule target set_logicalName newval
```

Vous pouvez utiliser `yCheckLogicalName()` pour vérifier si votre paramètre est valide. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval une chaîne de caractères représentant le nom logique du réveil agendé.

Retourne :

`YAPI_SUCCESS` si l'appel se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_minutes() YWakeUpSchedule
wakeupschedule→setMinutes()

Modifie toutes les minutes où un réveil doit avoir lieu

```
js    function set_minutes( bitmap)
cpp   int set_minutes( s64 bitmap)
m     -(int) setMinutes : (s64) bitmap
pas   LongInt set_minutes( bitmap: int64): LongInt
vb    function set_minutes( ) As Integer
cs    int set_minutes( long bitmap)
dnp   int set_minutes( long bitmap)
java  int set_minutes( long bitmap)
uwp   async Task<int> set_minutes( long bitmap)
py    set_minutes( bitmap)
php   function set_minutes( $bitmap)
es    async set_minutes( bitmap)
cmd   YWakeUpSchedule target set_minutes bitmap
```

Paramètres :

bitmap Minutes 00-59 de chaque heure où le réveil est actif.

Retourne :

YAPI_SUCCESS si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_minutesA() YWakeUpSchedule
wakeupschedule→setMinutesA()

Modifie les minutes de l'interval 00-29 où un réveil doit avoir lieu.

```
js    function set_minutesA( newval)
cpp   int set_minutesA( int newval)
m     -(int) setMinutesA : (int) newval
pas   integer set_minutesA( newval: LongInt): integer
vb    function set_minutesA( ByVal newval As Integer) As Integer
cs    int set_minutesA( int newval)
dnp   int set_minutesA( int newval)
java  int set_minutesA( int newval)
uwp   async Task<int> set_minutesA( int newval)
py    set_minutesA( newval)
php   function set_minutesA( $newval)
es    async set_minutesA( newval)
cmd   YWakeUpSchedule target set_minutesA newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant les minutes de l'interval 00-29 où un réveil doit avoir lieu

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_minutesB() YWakeUpSchedule
wakeupschedule→setMinutesB()

Modifie les minutes de l'interval 30-59 où un réveil doit avoir lieu.

```
js    function set_minutesB( newval)
cpp   int set_minutesB( int newval)
m     -(int) setMinutesB : (int) newval
pas   integer set_minutesB( newval: LongInt): integer
vb    function set_minutesB( ByVal newval As Integer) As Integer
cs    int set_minutesB( int newval)
dnp   int set_minutesB( int newval)
java  int set_minutesB( int newval)
uwp   async Task<int> set_minutesB( int newval)
py    set_minutesB( newval)
php   function set_minutesB( $newval)
es    async set_minutesB( newval)
cmd   YWakeUpSchedule target set_minutesB newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant les minutes de l'interval 30-59 où un réveil doit avoir lieu

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_monthDays() YWakeUpSchedule
wakeupschedule→setMonthDays()

Modifie les jours du mois où un réveil doit avoir lieu.

```
js    function set_monthDays( newval)
cpp   int set_monthDays( int newval)
m     -(int) setMonthDays : (int) newval
pas   integer set_monthDays( newval: LongInt): integer
vb    function set_monthDays( ByVal newval As Integer) As Integer
cs    int set_monthDays( int newval)
dnp   int set_monthDays( int newval)
java  int set_monthDays( int newval)
uwp   async Task<int> set_monthDays( int newval)
py    set_monthDays( newval)
php   function set_monthDays( $newval)
es    async set_monthDays( newval)
cmd   YWakeUpSchedule target set_monthDays newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant les jours du mois où un réveil doit avoir lieu

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_months()
wakeupschedule→setMonths()

YWakeUpSchedule

Modifie les mois où un réveil doit avoir lieu.

```
js    function set_months( newval)
cpp   int set_months( int newval)
m     -(int) setMonths : (int) newval
pas   integer set_months( newval: LongInt): integer
vb    function set_months( ByVal newval As Integer) As Integer
cs    int set_months( int newval)
dnp   int set_months( int newval)
java  int set_months( int newval)
uwp   async Task<int> set_months( int newval)
py    set_months( newval)
php   function set_months( $newval)
es    async set_months( newval)
cmd   YWakeUpSchedule target set_months newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant les mois où un réveil doit avoir lieu

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→set_userData()
wakeupschedule→setUserData()

YWakeUpSchedule

Enregistre un contexte libre dans l'attribut userData de la fonction, afin de le retrouver plus tard à l'aide de la méthode `get_userData`.

```
js      function set_userData( data)
cpp     void set_userData( void * data)
m       -(void) setUserData : (id) data
pas     set_userData( data: Tobject)
vb      procedure set_userData( ByVal data As Object)
cs      void set_userData( object data)
java    void set_userData( Object data)
py      set_userData( data)
php     function set_userData( $data)
es      async set_userData( data)
```

Cet attribut n'es pas utilisé directement par l'API. Il est à la disposition de l'appelant pour stocker un contexte.

Paramètres :

data objet quelconque à mémoriser

wakeupschedule→set_weekDays() YWakeUpSchedule
wakeupschedule→setWeekDays()

Modifie les jours de la semaine où un réveil doit avoir lieu.

```
js    function set_weekDays( newval)
cpp   int set_weekDays( int newval)
m     -(int) setWeekDays : (int) newval
pas   integer set_weekDays( newval: LongInt): integer
vb    function set_weekDays( ByVal newval As Integer) As Integer
cs    int set_weekDays( int newval)
dnp   int set_weekDays( int newval)
java  int set_weekDays( int newval)
uwp   async Task<int> set_weekDays( int newval)
py    set_weekDays( newval)
php   function set_weekDays( $newval)
es    async set_weekDays( newval)
cmd   YWakeUpSchedule target set_weekDays newval
```

N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Paramètres :

newval un entier représentant les jours de la semaine où un réveil doit avoir lieu

Retourne :

`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→unmuteValueCallbacks() YWakeUpSchedule

Réactive l'envoi de chaque changement de la valeur publiée au hub parent.

js	function **unmuteValueCallbacks**( )
cpp	int **unmuteValueCallbacks**( )
m	-(int) **unmuteValueCallbacks**
pas	LongInt **unmuteValueCallbacks**( ): LongInt
vb	function **unmuteValueCallbacks**( ) As Integer
cs	int **unmuteValueCallbacks**( )
dnp	int **unmuteValueCallbacks**( )
java	int **unmuteValueCallbacks**( )
uwp	async Task<int> **unmuteValueCallbacks**( )
py	**unmuteValueCallbacks**( )
php	function **unmuteValueCallbacks**( )
es	async **unmuteValueCallbacks**( )
cmd	YWakeUpSchedule **target unmuteValueCallbacks**

Cette fonction annule un précédent appel à `muteValueCallbacks()`. N'oubliez pas d'appeler la méthode `saveToFlash()` du module si le réglage doit être préservé.

Retourne :
`YAPI_SUCCESS` si l'opération se déroule sans erreur.

En cas d'erreur, déclenche une exception ou retourne un code d'erreur négatif.

wakeupschedule→wait_async() YWakeUpSchedule

Attend que toutes les commandes asynchrones en cours d'exécution sur le module soient terminées, et appelle le callback passé en paramètre.

```
js  function wait_async( callback, context)
es  wait_async( callback, context)
```

La fonction callback peut donc librement utiliser des fonctions synchrones ou asynchrones, sans risquer de bloquer la machine virtuelle Javascript.

Paramètres :

callback fonction de callback qui sera appelée dès que toutes les commandes en cours d'exécution sur le module seront terminées La fonction callback reçoit deux arguments: le contexte fourni par l'appelant et l'objet fonction concerné.

context contexte fourni par l'appelant, et qui sera passé tel-quel à la fonction de callback

Retourne :

rien du tout.

9. Problèmes courants

9.1. Par où commencer ?

Si c'est la première fois que vous utilisez un module Yoctopuce et ne savez pas trop par où commencer, allez donc jeter un coup d'œil sur le blog de Yoctopuce. Il y a une section dédiée aux débutants [1].

9.2. Linux et USB

Pour fonctionner correctement sous Linux la librairie a besoin d'avoir accès en écriture à tous les périphériques USB Yoctopuce. Or, par défaut, sous Linux les droits d'accès des utilisateurs non-root à USB sont limités à la lecture. Afin d'éviter de devoir lancer les exécutables en tant que root, il faut créer une nouvelle règle *udev* pour autoriser un ou plusieurs utilisateurs à accéder en écriture aux périphériques Yoctopuce.

Pour ajouter une règle *udev* à votre installation, il faut ajouter un fichier avec un nom au format "`##-nomArbitraire.rules`" dans le répertoire "`/etc/udev/rules.d`". Lors du démarrage du système, *udev* va lire tous les fichiers avec l'extension "`.rules`" de ce répertoire en respectant l'ordre alphabétique (par exemple, le fichier "`51-custom.rules`" sera interprété APRES le fichier "`50-udev-default.rules`").

Le fichier "`50-udev-default`" contient les règles *udev* par défaut du système. Pour modifier le comportement par défaut du système, il faut donc créer un fichier qui commence par un nombre plus grand que 50, qui définira un comportement plus spécifique que le défaut du système. Notez que pour ajouter une règle vous aurez besoin d'avoir un accès root sur le système.

Dans le répertoire `udev_conf` de l'archive du *VirtualHub*[2] pour Linux, vous trouverez deux exemples de règles qui vous éviterons de devoir partir de rien.

Exemple 1: 51-yoctopuce.rules

Cette règle va autoriser tous les utilisateurs à accéder en lecture et en écriture aux périphériques Yoctopuce USB. Les droits d'accès pour tous les autres périphériques ne seront pas modifiés. Si ce scénario vous convient il suffit de copier le fichier "`51-yoctopuce_all.rules`" dans le répertoire "`/etc/udev/rules.d`" et de redémarrer votre système.

[1] voir: http://www.yoctopuce.com/FR/blog_by_categories/pour-les-debutants
[2] http://www.yoctopuce.com/EN/virtualhub.php

```
# udev rules to allow write access to all users
# for Yoctopuce USB devices
SUBSYSTEM=="usb", ATTR{idVendor}=="24e0", MODE="0666"
```

Exemple 2: 51-yoctopuce_group.rules

Cette règle va autoriser le groupe "yoctogroup" à accéder en lecture et écriture aux périphériques Yoctopuce USB. Les droits d'accès pour tous les autres périphériques ne seront pas modifiés. Si ce scénario vous convient il suffit de copier le fichier "`51-yoctopuce_group.rules`" dans le répertoire "`/etc/udev/rules.d`" et de redémarrer votre système.

```
# udev rules to allow write access to all users of "yoctogroup"
# for Yoctopuce USB devices
SUBSYSTEM=="usb", ATTR{idVendor}=="24e0", MODE="0664", GROUP="yoctogroup"
```

9.3. Plateformes ARM: HF et EL

Sur ARM il existe deux grandes familles d'executables: HF (Hard Float) et EL (EABI Little Endian). Ces deux familles ne sont absolument pas compatibles entre elles. La capacité d'une machine ARM à faire tourner des exécutables de l'une ou l'autre de ces familles dépend du hardware et du système d'exploitation. Les problèmes de compatibilité entre ArmHL et ArmEL sont assez difficiles à diagnostiquer, souvent même l'OS se révèle incapable de distinguer un exécutable HF d'un exécutable EL.

Tous les binaires Yoctopuce pour ARM sont fournis pré-compilée pour ArmHF et ArmEL, si vous ne savez à quelle famille votre machine ARM apartient, essayez simplement de lancer un exécutable de chaque famille.

9.4. Les exemples de programmation n'ont pas l'air de marcher

La plupart des exemples de programmation de l'API Yoctopuce sont des programmes en ligne de commande et ont besoin de quelques paramètres pour fonctionner. Vous devez les lancer depuis l'invite de commande de votre système d'exploitation ou configurer votre IDE pour qu'il passe les paramètres corrects au programme [3].

9.5. Module alimenté mais invisible pour l'OS

Si votre YoctoHub-GSM-3G-NA est branché par USB et que sa LED bleue s'allume, mais que le module n'est pas vu par le système d'exploitation, vérifiez que vous utilisez bien un vrai câble USB avec les fils pour les données, et non pas un câble de charge. Les câbles de charge n'ont que les fils d'alimentation.

9.6. Another process named xxx is already using yAPI

Si lors de l'initialisation de l'API Yoctopuce, vous obtenez le message d'erreur "*Another process named xxx is already using yAPI*", cela signifie qu'une autre application est déjà en train d'utiliser les modules Yoctopuce USB. Sur une même machine, un seul processus à la fois peut accéder aux modules Yoctopuce par USB. Cette limitation peut facilement être contournée en utilisant un VirtualHub et le mode réseau [4].

[3] voir: http://www.yoctopuce.com/FR/article/a-propos-des-programmes-d-exemples
[4] voir: http://www.yoctopuce.com/FR/article/message-d-erreur-another-process-is-already-using-yapi

9.7. Déconnexions, comportement erratique

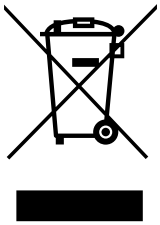
Si votre YoctoHub-GSM-3G-NA se comporte de manière erratique et/ou se déconnecte du bus USB sans raison apparente, vérifiez qu'il est alimenté correctement. Evitez les câbles d'une longueur supérieure à 2 mètres. Au besoin, intercalez un hub USB alimenté [5] [6].

9.8. Impossible de contacter les sous-devices par USB

Le but du YoctoHub-GSM-3G-NA est de fournir une connectivitÃ© rÃ©seau aux sous-modules qui lui sont connectÃ©, il ne se comporte pas comme un hub USB. Le port USB du YoctoHub-GSM-3G-NA ne sert qu'Ã l'alimenter et le configurer. Pour accÃ©der aux modules connectÃ©s au hub, vous devez impÃ©rativement passer par une connexion rÃ©seau.

9.9. Module endommagé

Yoctopuce s'efforce de réduire la production de déchets électroniques. Si vous avez l'impression que votre YoctoHub-GSM-3G-NA ne fonctionne plus, commencez par contacter le support Yoctopuce par e-mail pour poser un diagnostic. Même si c'est suite à une mauvaise manipulation que le module a été endommagé, il se peut que Yoctopuce puisse le réparer, et ainsi éviter de créer un déchet électronique.



Déchets d'équipements électriques et électroniques (DEEE) Si voulez vraiment vous débarasser de votre YoctoHub-GSM-3G-NA, ne le jetez pas à la poubelle, mais ramenez-le à l'un des points de collecte proposé dans votre région afin qu'il soit envoyé à un centre de recyclage ou de traitement spécialisé.

[5] voir: http://www.yoctopuce.com/FR/article/cables-usb-la-taille-compte
[6] voir: http://www.yoctopuce.com/FR/article/combien-de-capteurs-usb-peut-on-connecter

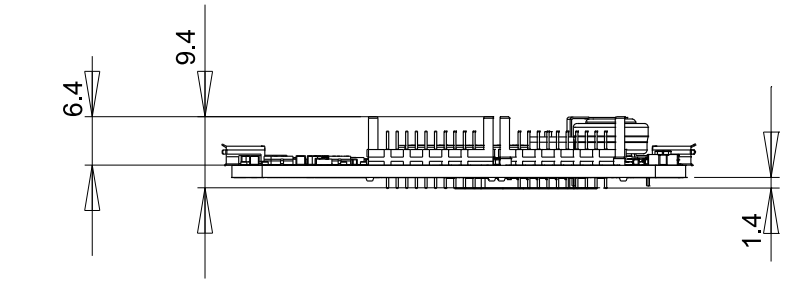
10. Caractéristiques

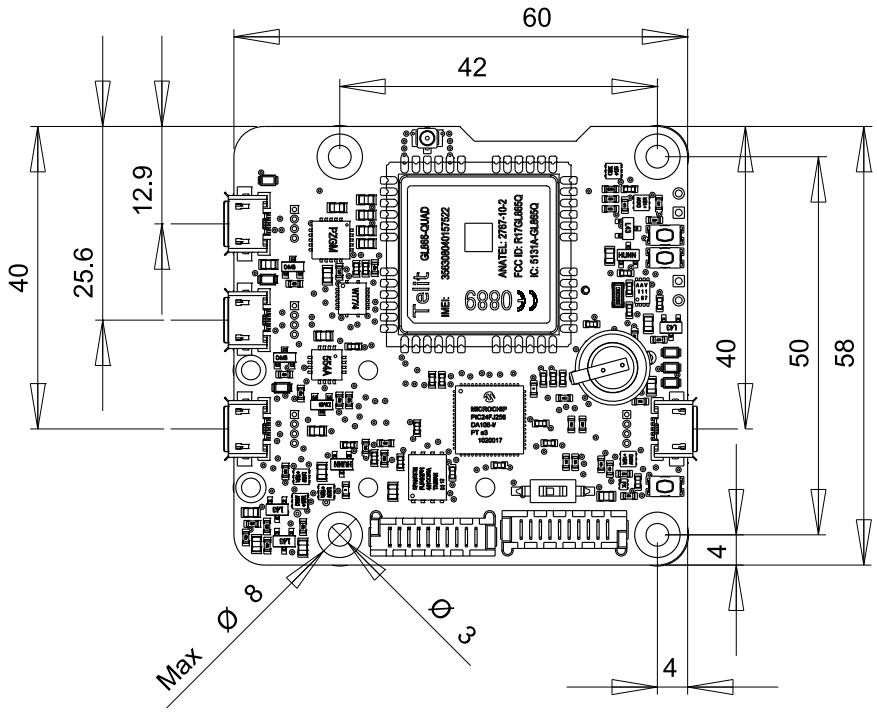
Vous trouverez résumées ci-dessous les principales caractéristiques techniques de votre module YoctoHub-GSM-3G-NA

Identifiant produit	YHUBGSM4
Révision matérielle†	Rev. C
Connecteur USB	micro-B
Epaisseur	9.5 mm
Largeur	58 mm
Longueur	60 mm
Poids	34 g
Chipset	Telit UL865-NAD
Fréquence	850 et 1900 MHz
Classe de protection selon IEC 61140	classe III
Temp. de fonctionnement normale	5...40 °C
Temp. de fonctionnement étendue‡	-20...70 °C
Consommation USB	100 mA
Conformité RoHS	RoHS III (2011/65/UE+2015/863)
USB Vendor ID	0x24E0
USB Device ID	0x0061
Boîter recommandé	YoctoBox-HubWlan-Transp
Code tarifaire harmonisé	8542.3190
Fabriqué en	Suisse

† Ces spécifications correspondent à la révision matérielle actuelle du produit. Les spécifications des versions antérieures peuvent être inférieures.

‡ La plage de température étendue est définie d'après les spécifications des composants et testée sur une durée limitée (1h). En cas d'utilisation prolongée hors de la plage de température standard, il est recommandé procéder à des tests extensifs avant la mise en production.





All dimensions are in mm
Toutes les dimensions sont en mm

YoctoHub-GSM

A4

Scale
1:1
Echelle