



YoctoHub-Wireless-SR

Mode d'emploi

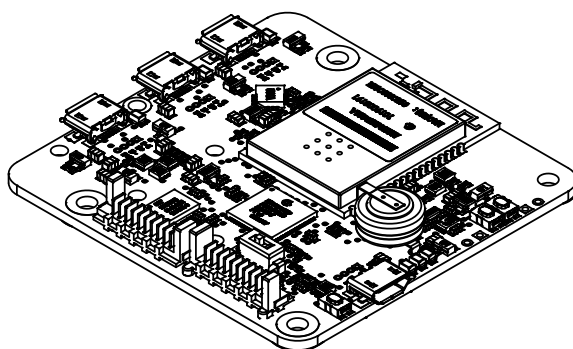
Table des matières

1. Introduction	1
2. Présentation	3
2.1. Les éléments du YoctoHub-Wireless-SR	3
3. Premiers pas	7
3.1. Configuration manuelle	7
3.2. Configuration automatisée	11
3.3. Connexions	11
4. Montage	15
4.1. Fixation	15
4.2. Fixation d'un sous-module	16
5. Utilisation du YoctoHub-Wireless-SR	17
5.1. Localisation des modules	17
5.2. Test des modules	18
5.3. Configuration des modules	18
5.4. Upgrades des firmwares	19
6. Contrôle d'accès	21
6.1. Accès "admin" protégé	22
6.2. Accès "user" protégé	22
6.3. Influence sur les API	23
6.4. Effacement des mots de passe	23
7. Interactions avec l'extérieur	25
7.1. Configuration	25
7.2. Emoncms	26
7.3. Valarm.net	26
7.5. InfluxDB	27
7.6. PRTG	27
7.7. MQTT	27

7.8. Yocto-API callback	27
7.9. User defined callback	28
8. Programmation	31
8.1. Accès aux modules connectés	31
8.2. Contrôle du YoctoHub-Wireless-SR	31
9. Mise en sommeil	33
9.1. Configuration manuelle du système de réveil	33
9.2. Paramétrage du système de réveil par logiciel	34
10. Personnalisation de l'interface Web	37
10.1. Utilisation	37
10.2. Limitations	38
11. Référence de l'API de haut niveau	39
11.1. La classe YHubPort	40
11.2. La classe YWireless	88
11.3. La classe YNetwork	144
11.4. La classe YFiles	258
11.5. La classe YRealTimeClock	310
11.6. La classe YWakeUpMonitor	359
11.7. La classe YWakeUpSchedule	417
12. Problèmes courants	483
12.1. Par où commencer ?	483
12.2. Linux et USB	483
12.3. Plateformes ARM: HF et EL	484
12.4. Les exemples de programmation n'ont pas l'air de marcher	484
12.5. Module alimenté mais invisible pour l'OS	484
12.6. Another process named xxx is already using yAPI	484
12.7. Déconnexions, comportement erratique	485
12.8. RegisterHub d'un VirtualHub déconnecte le précédent	485
12.9. Commandes ignorées	485
12.10. Impossible de contacter les sous-devices par USB	485
12.11. Network Readiness coincé à	485
12.12. Module endommagé	485
13. Caractéristiques	487

1. Introduction

Le YoctoHub-Wireless-SR est un module électronique de 60x58mm qui permet de contrôler d'autres modules Yoctopuce à travers une connexion réseau sans fil. Vu de l'extérieur, ce module se comporte exactement comme un ordinateur classique faisant tourner un VirtualHub¹: même interface, mêmes fonctionnalités.



Le YoctoHub-Wireless-SR

Le YoctoHub-Wireless-SR 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. Quelques réglages simples permettent son utilisation dans toutes sortes d'environnements réseau. Ces 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.

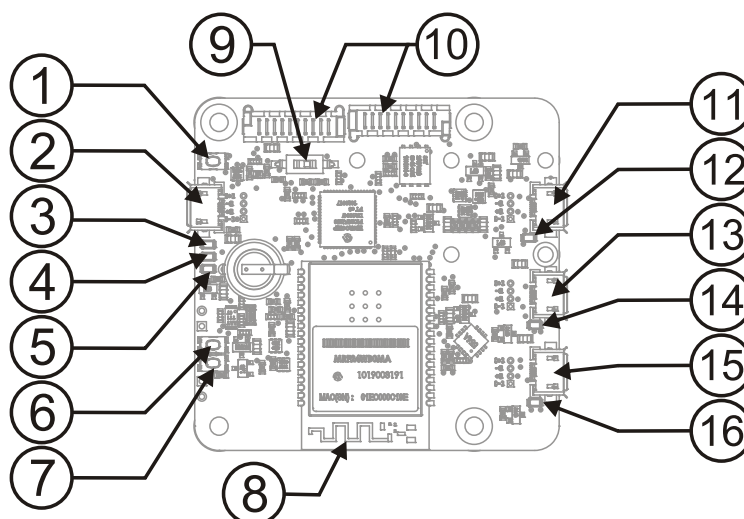
Le YoctoHub-Wireless-SR n'est pas un hub USB standard avec accès réseau. Bien qu'utilisant du câblage USB, ses ports descendants utilisent un protocole propriétaire, plus simple qu'USB. Il n'est par conséquent pas possible de contrôler, ni même d'alimenter, des périphériques USB standards avec un YoctoHub-Wireless-SR.

Yoctopuce vous remercie d'avoir fait l'acquisition de ce YoctoHub-Wireless-SR et espère sincèrement qu'il vous donnera entière satisfaction. Les ingénieurs Yoctopuce se sont donnés beaucoup de mal pour que votre YoctoHub-Wireless-SR soit facile à installer n'importe où et soit facile à utiliser en toutes circonstances. Néanmoins, si ce module venait à vous décevoir, n'hésitez pas à contacter le support Yoctopuce².

¹ <http://www.yoctopuce.com/FR/virtualhub.php>

² support@yoctopuce.com

2. Présentation



- | | |
|--|---|
| 1: Yocto-bouton | 9: Neutralisation de la mise en sommeil |
| 2: Port USB de contrôle + alimentation | 10: Connexion dorsale |
| 3: Yocto-Led | 11: Port descendant 1 |
| 4: Indicateur de sur-consommation | 12: Indicateur port descendant 1 |
| 5: Indicateur de transfert réseau | 13: Port descendant 2 |
| 6: Touche réveil | 14: Indicateur port descendant 2 |
| 7: Touche mise en sommeil | 15: Port descendant 3 |
| 8: Antenne | 16: Indicateur port descendant 3 |

2.1. Les éléments du YoctoHub-Wireless-SR

Le numéro de série

Chaque Yocto-module a un numéro de série unique attribué en usine, pour les modules YoctoHub-Wireless-SR ce numéro commence par YHUBWLN2. 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 morse¹. 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 Yoctopuce².

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

Ce connecteur permet d'alimenter le YoctoHub-Wireless-SR 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-Wireless-SR 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-Wireless-SR 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-Wireless-SR et le module Yoctopuce n'est pas de l'USB mais un protocole propriétaire plus léger. De ce fait le YoctoHub-Wireless-SR ne peut pas gérer des périphériques autres que des modules Yoctopuce. Un hub USB standard ne fonctionnera pas non plus³. Si vous désirez brancher plus de trois modules Yoctopuce, utilisez le connecteur dorsal pour y connecter un ou plusieurs YoctoHub-Shield⁴.

Attention, les connecteurs USB du YoctoHub-Wireless-SR 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 souder un fil USB directement dans les trous espacés de 1.27mm prévus à cet effet, près du connecteur.

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-Wireless-SR.

⁴ www.yoctopuce.com/FR/products/yoctohub-shield

Antenne intégrée

Le YoctoHub-Wireless-SR dispose d'une antenne intégrée, ce qui vous dispense d'utiliser une antenne externe. Du fait de cette antenne intégrée, l'orientation du YoctoHub-Wireless-SR va avoir une influence sur ses performances. Veillez à ne pas installer le YoctoHub-Wireless-SR dans un boîtier métallique.

Indicateur de sur-consommation

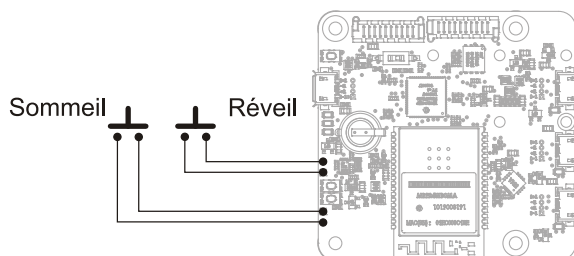
Le YoctoHub-Wireless-SR analyse en permanence sa consommation. S'il détecte une consommation globale de plus de 2A suite à une surcharge sur un des ports descendants par exemple, il va automatiquement désactiver tous les ports descendants et allumer l'indicateur de sur-consommation. Pour isoler la source 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-Wireless-SR 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 temps normal, le YoctoHub-Wireless-SR consomme environ 0,5 Watt (110mA), 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-Wireless-SR 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 μ W (15 μ A).

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-Wireless-SR. 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-Wireless-SR 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.

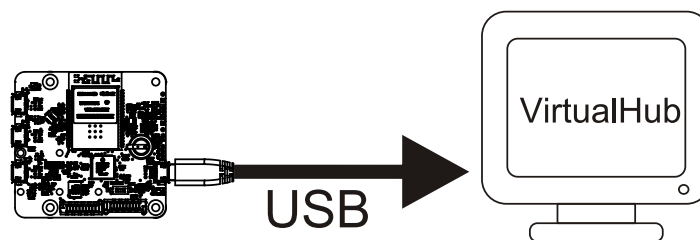
3. Premiers pas

Ce chapitre a pour but de vous aider à connecter et configurer votre YoctoHub-Wireless-SR pour la première fois

3.1. Configuration manuelle

Vous pouvez configurer votre YoctoHub-Wireless-SR 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-Wireless-SR. Vous aurez besoin d'un câble USB A-MicroB.



Configuration: raccordez par USB votre YoctoHub-Wireless-SR 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-Wireless-SR

Serial	Logical Name	Description	Action
VIRIHUB0-7d1a86fb0		VirtualHub	configure view log file
YHUBWLN1-11D70		YoctoHub-Wireless	configure view log file beacon

Liste des modules Yoctopuce raccordés par USB à votre ordinateur, dont votre YoctoHub-Wireless-SR

Cliquez sur le bouton **configure** correspondant à votre YoctoHub-Wireless-SR, vous obtiendrez la fenêtre de configuration du module. Cette fenêtre comporte une section **Network configuration**.

¹ <http://www.yoctopuce.com/FR/virtualhub.php>

YHUBWLN1-11D70

Edit parameters for device YHUBWLN1-11D70, and click on the **Save** button.

Serial # YHUBWLN1-11D70
 Product name: YoctoHub-Wireless
 Firmware: 12704 upgrade
 Logical name:
 Luminosity:  (signal leds only)

Device functions

Each function of the device has a physical name and a logical name. You can change the logical name using the **rename** button.

YHUBWLN1-11D70.files / rename
 User files: 0 file, 3724 KB available manage files
 YHUBWLN1-11D70.hubPort1 / YWATTMK1-0E24B rename
 YHUBWLN1-11D70.hubPort2 / rename
 YHUBWLN1-11D70.hubPort3 / RELAYHI1-00033 rename
 YHUBWLN1-11D70.network / YHUBWLN1-11D70 rename
 YHUBWLN1-11D70.realTimeClock / rename
 YHUBWLN1-11D70.wakeUpMonitor / rename
 YHUBWLN1-11D70.wakeUpSchedule1 / rename
 YHUBWLN1-11D70.wakeUpSchedule2 / rename
 YHUBWLN1-11D70.wireless / rename

Wake-up Scheduler

Maximum power-on duration: no limit edit
 Next occurrence of wake-up schedule 1: Not set setup
 Next occurrence of wake-up schedule 2: Not set setup

Network configuration (4- WWW ready)

WLAN settings: SWEETSHORTWPA edit
 Device name: YHUBWLN1-11D70 edit
 IP addressing: Automatic by DHCP edit
 (current IP: 192.168.1.54)

Incoming connections

Authentication to read information from the devices: NO edit
 Authentication to make changes to the devices: NO edit

Outgoing callbacks

Callback URL: http://www.stuckelberg.ch/testcb/doublebuff.php edit
 Callback method: POST Yocto-API
 Delay between callbacks: min: 1 [s] max: 30 [s]
 Network downtime to reboot: no downtime limit edit

Save Cancel

Fenêtre de configuration du module YoctoHub-Wireless-SR

Connexion au réseau sans fil

La première chose à faire consiste à configurer votre YoctoHub-Wireless-SR pour qu'il se connecte à votre réseau WiFi. Pour cela cliquez sur le bouton **edit** correspondant à **WLAN settings** dans la section **Network configuration**, la fenêtre de configuration du réseau sans fil apparaît:

Fenêtre de configuration du réseau sans fil.

Vous pouvez alors choisir si vous souhaitez connecter votre YoctoHub-Wireless-SR à un réseau existant, ou si vous préférez entrer manuellement le SSID du réseau que vous voulez utiliser.

Vous pouvez aussi configurer le YoctoHub-Wireless-SR pour qu'il génère son propre réseau sans fil en mode *ad-hoc*. Vous pourrez alors vous connecter directement sur le YoctoHub-Wireless-SR sans avoir à passer par un serveur d'infrastructure (point d'accès). Soyez toutefois conscients que ce mode de fonctionnement *ad-hoc* a des limitations importantes par rapport à un vrai réseau WiFi. En particulier, les appareils sous Android ne sont pas capable de s'y connecter.

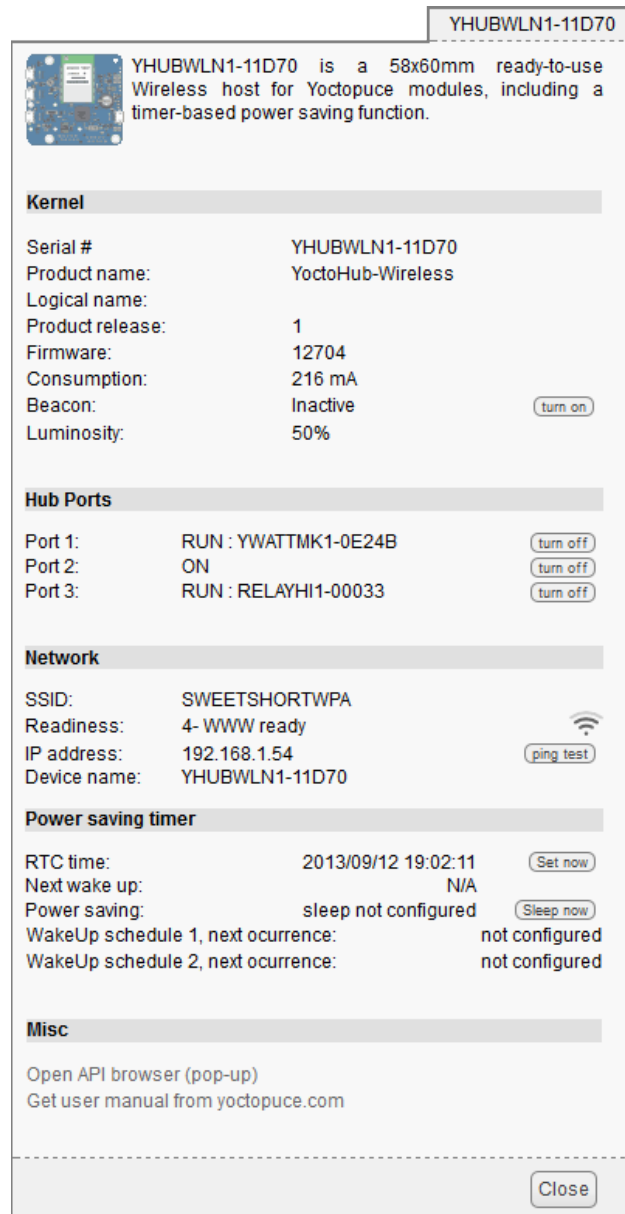
Après avoir entré les paramètres de réseau sans fil et éventuellement les avoir testés, vous pouvez cliquer sur le bouton **Ok** pour fermer cette fenêtre de configuration et retourner à la fenêtre de configuration générale.

Vous pouvez aussi, en cas de besoin, configurer quelle adresse IP doit être attribuée au YoctoHub-Wireless-SR. Pour ce faire, cliquez sur le bouton **edit** en face de la ligne **ip adresssing**.

Vous pouvez choisir si l'adresse IP de votre YoctoHub-Wireless-SR doit être attribuée par DHCP ou si elle doit être fixe. L'option DHCP est recommandée dans la mesure où cette fonctionnalité est supportée par la plupart des boîtiers ADSL (c'est la configuration par défaut). Si vous ne savez pas ce qu'est un serveur DHCP mais avez l'habitude de brancher des appareils sur votre réseau et de les voir marcher sans problème, ne touchez à rien.

Vous pouvez aussi choisir le nom réseau de votre YoctoHub-Wireless-SR. Vous pourrez ainsi accéder à votre YoctoHub-Wireless-SR en utilisant ce nom plutôt que son adresse IP. Une fois la partie réseau configurée, cliquez sur le bouton **Save**. Cela aura pour effet de sauvegarder vos modifications et de fermer la fenêtre de configuration. Ces modifications étant sauvegardées dans la mémoire persistante du YoctoHub-Wireless-SR, il s'en rappellera même après avoir été privé de courant.

Cliquez sur le numéro de série correspondant à votre YoctoHub-Wireless-SR. Cela ouvrira la fenêtre détails de votre module:



Les propriétés du YoctoHub-Wireless-SR

Cette fenêtre comporte une section qui relate l'état de la partie réseau du YoctoHub-Wireless-SR. 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- search for link: Le module cherche une liaison réseau. Si cet état persiste, il est probable que le réseau wifi recherché n'est pas à proximité
- 1- network exists: le réseau wifi recherché a été détecté
- 2- network linked: le YoctoHub-Wireless-SR a pu se connecter au réseau.
- 3- LAN ready: le réseau local est opérationnel (adresse IP obtenue)
- 4- WWW ready: le module a pu vérifier la connectivité à Internet en se connectant à un serveur de temps (NTP).

Après avoir vérifié que votre module a bien une adresse IP valide, vous pouvez fermer la fenêtre détails, arrêter votre VirtualHub et débrancher le câble USB de contrôle de votre ordinateur: il suffira que le module soit alimenté pour l'utiliser.

Vous pouvez désormais accéder à votre YoctoHub-Wireless-SR en tapant directement son adresse IP dans la barre d'adresse de votre browser favori. Le module répond au port HTTP standard, mais aussi au port 4444 utilisé par le VirtualHub. Si l'adresse IP de votre module est `192.168.0.10`, vous pourrez donc le joindre avec l'URL `http://192.168.0.10`.

Vous obtiendrez alors directement l'interface du YoctoHub-Wireless-SR. Cette interface est en tout point identique à celle du *VirtualHub*. Vous retrouvez le YoctoHub-Wireless-SR sur la première ligne et les modules connectés au YoctoHub-Wireless-SR sur les suivantes.

Serial	Logical Name	Description	Action
YHUBWLN1-11D70		YoctoHub-Wireless	configure view log file beacon
YWATTMK1-0E24B		Yocto-Watt	configure view log file beacon
RELAYHI1-00033		Yocto-PowerRelay	configure view log file beacon

L'interface du YoctoHub-Wireless-SR est identique à celle d'un VirtualHub.

Si vous avez attribué un nom à votre YoctoHub-Wireless-SR, vous pouvez aussi utiliser ce nom sur le réseau local. Par exemple, si vous avez utilisé le nom réseau *yoctohub*, vous pouvez contacter le module avec l'URL <http://yoctohub> sous Windows et avec l'URL <http://yoctohub.local> sous Mac OS X et Linux. Notez que cette technique est limitée au sous-réseau du YoctoHub-Wireless-SR. Si vous voulez contacter le module par nom depuis un autre réseau, vous devez utiliser une infrastructure DNS classique.

3.2. Configuration automatisée

Il est possible d'industrialiser la configuration réseau du YoctoHub-Wireless-SR. 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 commande².

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

3.3. Connexions

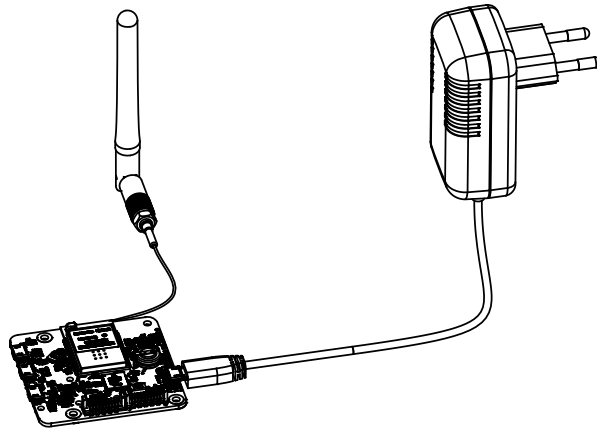
Alimentation

Le YoctoHub-Wireless-SR 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-Wireless-SR consomme environ 110mA, auxquels il faudra ajouter la consommation de chaque sous-module. Le YoctoHub-Wireless-SR 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.

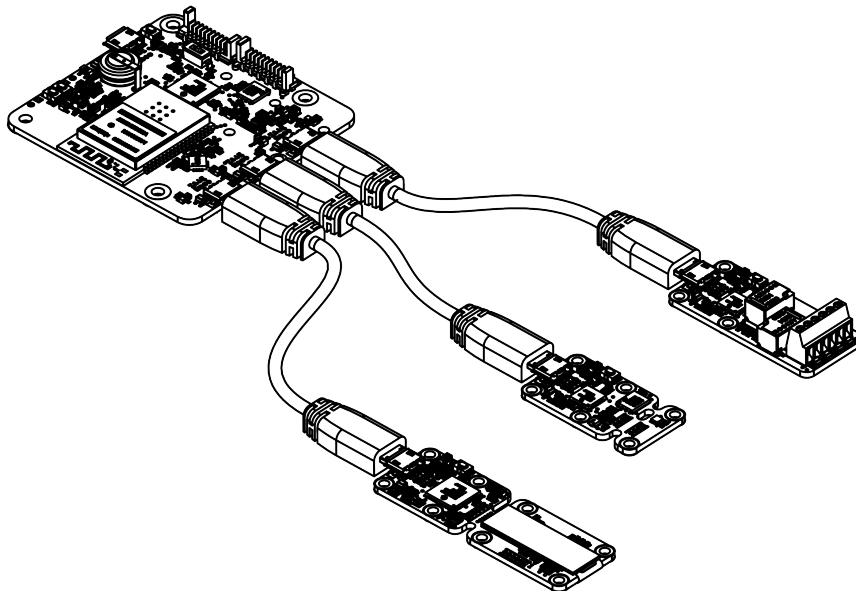
² <http://www.yoctopuce.com/FR/libraries.php>



Le YoctoHub-Wireless-SR peut être alimenté par un chargeur USB

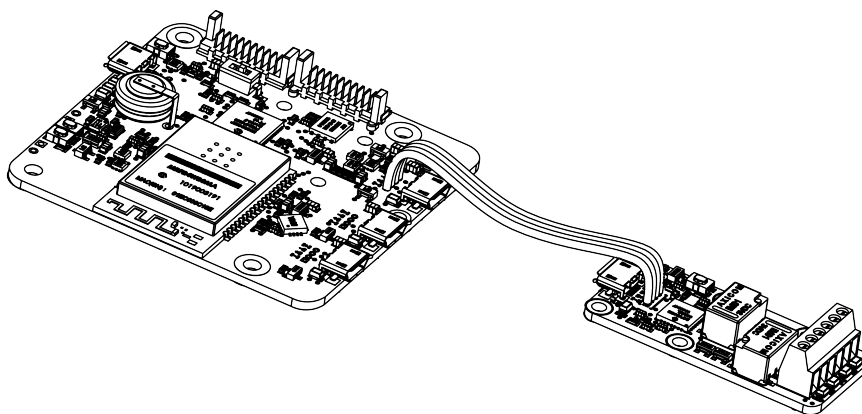
Sous-modules

Le YoctoHub-Wireless-SR est capable de piloter tous les modules Yoctopuce de la gamme *Yocto*. Ces modules peuvent être 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.



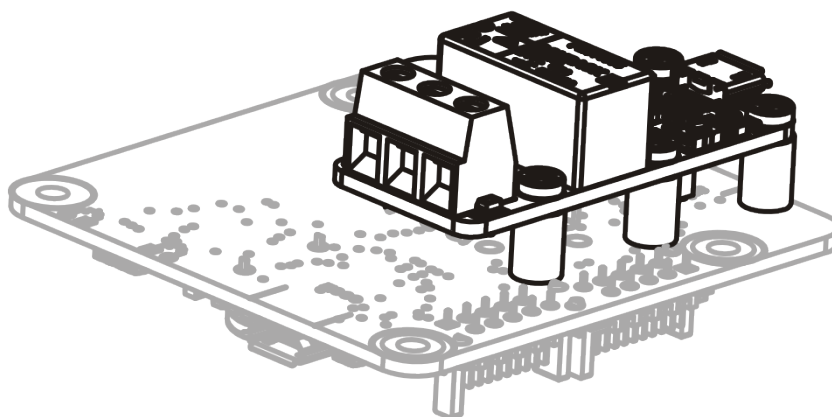
Connexion des sous-modules à l'aide de câbles USB

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-Wireless-SR, 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-Wireless-SR est conçu pour que vous puissiez fixer un module simple largeur directement dessus. Vous aurez besoin de vis, d'entretoises³ et d'un connecteur au pas 1.27mm⁴. Vous pouvez ainsi transformer un module Yoctopuce USB en module réseau tout en gardant un format très compact.



Fixation d'un module directement sur le hub

Attention, le YoctoHub-Wireless-SR est conçu pour piloter des modules Yoctopuce uniquement. En effet le protocole utilisé entre le YoctoHub-Wireless-SR 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-Wireless-SR, le port en question sera automatiquement désactivé pour éviter d'endommager le périphérique.

³ <http://www.yoctopuce.com/FR/products/accessoires-et-connectique/fix-2-5mm>

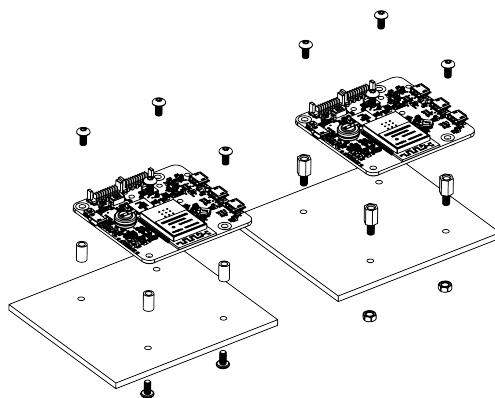
⁴ <http://www.yoctopuce.com/FR/products/accessoires-et-connectique/board2board-127>

4. Montage

Ce chapitre fournit des explications importantes pour utiliser votre module YoctoHub-Wireless-SR 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.



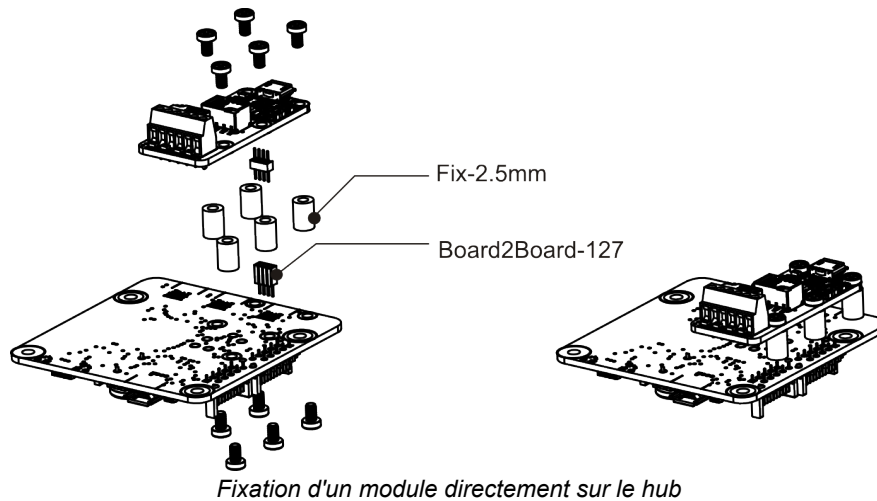
Exemples de montage sur un support.

Le module YoctoHub-Wireless-SR 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-Wireless-SR, 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-Wireless-SR 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-Wireless-SR 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 utiliser un simple câble USB MicroB-MicroB, OTG ou non.



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-Wireless-SR, et le contact carré sur le module doit être connecté au contact carré sur le port descendant du YoctoHub-Wireless-SR. 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-Wireless-SR 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. Utilisation du YoctoHub-Wireless-SR

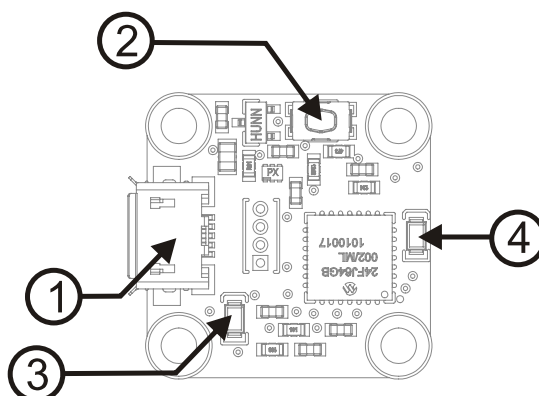
Outre fournir un accès réseau aux module Yoctopuce, le YoctoHub-Wireless-SR permet de tester et configurer vos modules Yoctopuce. Pour ce faire connectez-vous sur votre YoctoHub-Wireless-SR à l'aide de votre navigateur internet favori¹. Utilisez l'adresse IP du YoctoHub-Wireless-SR ou encore son nom réseau, par exemple `http://192.168.0.10`. Une liste comprenant votre YoctoHub-Wireless-SR ainsi que les modules qui lui sont connectés devrait apparaître.

Serial	Logical Name	Description	Action
YHUBWLN1-11D70		YoctoHub-Wireless	configure view log file beacon
MAXII001-121FD		Yocto-Maxi-IO	configure view log file beacon
RELAYHI1-00033		Yocto-PowerRelay	configure view log file beacon

Interface Web du YoctoHub-Wireless-SR.

5.1. Localisation des modules

L'interface principale vous montre une ligne par module connecté, si vous avez plusieurs modules du même modèle, vous pouvez localiser un module particulier en cliquant sur le bouton **beacon** correspondant: cela aura pour effet de faire clignoter la Led bleue du module et d'afficher sur l'interface une pastille bleue au début de la ligne correspondante. Vous pouvez faire la même manipulation en appuyant sur le Yocto-bouton d'un module connecté.

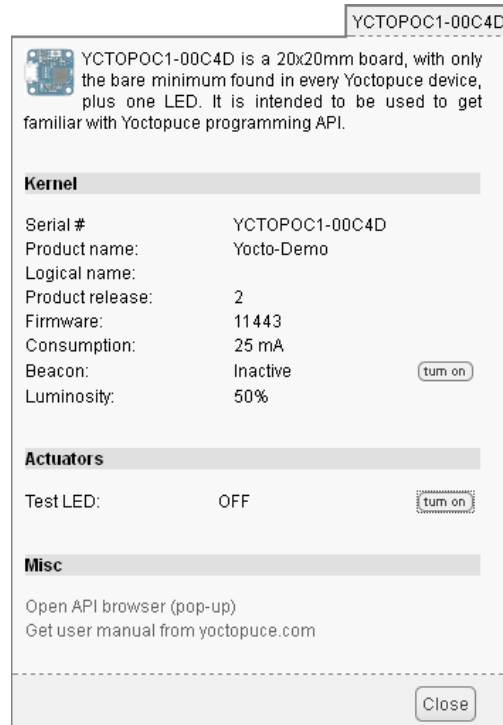


Yocto-bouton(1) et Led(2) de localisation d'un module Yocto-Demo. Ces deux éléments sont généralement placés au même endroit sur les modules.

¹ L'interface du YoctoHub-Wireless-SR est régulièrement testée sur Internet Explorer, Firefox, Chrome et Safari. Elle ne fonctionne pas avec Opéra

5.2. Test des modules

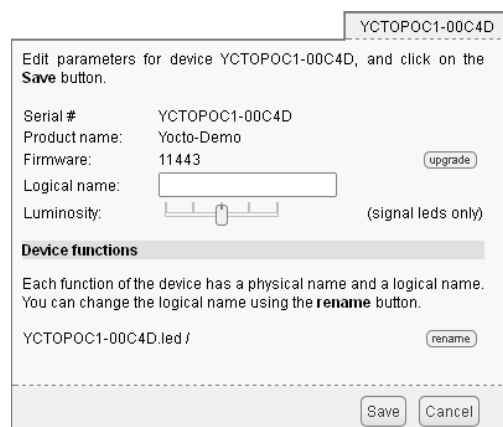
Pour tester un module, cliquez simplement sur le numéro de série d'un module dans l'interface, une fenêtre spécifique au module s'ouvrira. Cette fenêtre permet généralement d'activer les fonctions principales du module. Reportez-vous au manuel du module correspondant pour plus de détails.²



Fenêtre "détails" du module Yocto-Demo, obtenue via l'interface du YoctoHub-Wireless-SR.

5.3. Configuration des modules

Vous pouvez configurer un module en cliquant sur le bouton **Configure** correspondant dans l'interface principale, une fenêtre spécifique au module s'ouvre alors. Cette fenêtre permet au minimum de donner un nom logique au module ainsi que de mettre à jour son firmware. Reportez-vous au manuel du module correspondant pour plus de détails.



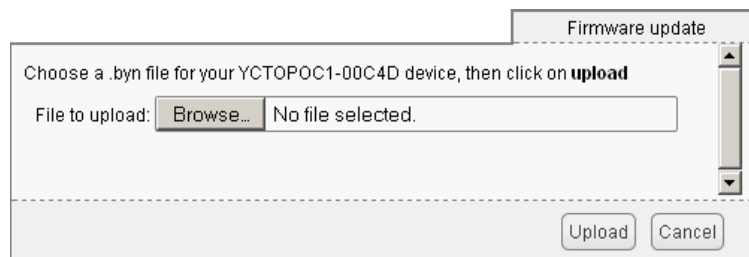
Fenêtre "configure" du module Yocto-Demo.

² Vous n'êtes pas obligé d'avoir un YoctoHub-Wireless-SR plus récent qu'un module pour le tester/configurer: tous les éléments spécifiques aux interfaces des modules sont stockés dans la ROM des modules, et non pas dans le YoctoHub-Wireless-SR.

5.4. Upgrades des firmwares

Les modules Yoctopuce sont en fait de véritables ordinateurs, ils contiennent même un petit serveur Web. Et comme tous les ordinateurs, il est possible de mettre à jour leur logiciel de contrôle (firmware). Des nouveaux firmwares pour chaque module sont régulièrement publiés, ils permettent généralement d'ajouter de nouvelles fonctionnalités au module, et/ou de corriger d'éventuels bugs³.

Pour mettre à jour le firmware d'un module, vous devez d'abord vous procurer le firmware, il peut être téléchargé depuis la page produit du module sur le site de Yoctopuce⁴. L'interface propose aussi un lien direct si elle détecte que le firmware n'est pas à jour⁵. Ces firmwares se présentent sous la forme de fichiers `.byn` de quelques dizaines de Kilo-octets, sauvez celui qui vous intéresse sur votre disque local.



Fenêtre de mise à jour du firmware.

Une fois votre fichier de firmware disponible localement, ouvrez la fenêtre **configuration** d'un module et cliquez sur le bouton **upgrade**. L'interface va vous demander de choisir le fichier de firmware que vous désirez utiliser. Entrez le nom du fichier et cliquez sur **Upload**. A partir de là, tout est automatique, le YoctoHub-Wireless-SR va faire redémarrer le module en mode "mise à jour", mettre à jour le firmware, puis redémarrer le module en mode normal. Les réglages de configuration du module seront préservés. Ne débranchez pas le module pendant la procédure de mise à jour.

Le firmware du YoctoHub-Wireless-SR peut être mis à jour de la même manière

En cas de perte de contrôle pendant une mise à jour du firmware (panne de courant ou débranchement involontaire), il est toujours possible de forcer manuellement un rechargement du firmware, même si le sous-module n'apparaît même plus dans la fenêtre du YoctoHub-Wireless-SR. Pour ce faire, débranchez le sous-module et rebranchez-le en maintenant le Yocto-bouton pressé. Le module démarrera alors dans le mode de mise à jour du firmware, et vous pourrez recommencer la procédure de chargement.

³ Ne faites jamais confiance à des gens qui vous disent que leur logiciel n'a pas de bug :-)

⁴ www.yoctopuce.com

⁵ A condition qu'elle ait réussi à accéder au site Web de Yoctopuce

6. Contrôle d'accès

Le YoctoHub-Wireless-SR vous permet d'instaurer un contrôle d'accès à vos modules Yoctopuce. Pour ce faire, cliquez simplement sur le bouton **Configure** de la ligne correspondant au YoctoHub-Wireless-SR dans l'interface.

Serial	Logical Name	Description	Action
YHUEWLN1-11D70		YoctoHub-Wireless	configure view log file beacon
MAXII001-121FD		Yocto-Maxi-IO	configure view log file beacon
RELAYHI1-00033		Yocto-PowerRelay	configure view log file beacon

Show device functions

Cliquez sur le bouton "Configure" de la première ligne

Cela aura pour effet de faire apparaître la fenêtre de configuration du YoctoHub-Wireless-SR.

YHUBWLN1-11D70

Edit parameters for device YHUBWLN1-11D70, and click on the **Save** button.

Serial #: YHUBWLN1-11D70
 Product name: YoctoHub-Wireless
 Firmware: 12704 upgrade
 Logical name:
 Luminosity: (signal leds only)

Device functions

Each function of the device has a physical name and a logical name. You can change the logical name using the **rename** button.

YHUBWLN1-11D70.files / rename
 User files: 0 file, 3724 KB available manage files
 YHUBWLN1-11D70.hubPort1 / YWATMK1-0E24B rename
 YHUBWLN1-11D70.hubPort2 / rename
 YHUBWLN1-11D70.hubPort3 / RELAYHI1-00033 rename
 YHUBWLN1-11D70.network / YHUBWLN1-11D70 rename
 YHUBWLN1-11D70.realTimeClock / rename
 YHUBWLN1-11D70.wakeUpMonitor / rename
 YHUBWLN1-11D70.wakeUpSchedule1 / rename
 YHUBWLN1-11D70.wakeUpSchedule2 / rename
 YHUBWLN1-11D70.wireless / rename

Wake-up Scheduler

Maximum power-on duration: no limit edit
 Next occurrence of wake-up schedule 1: Not set setup
 Next occurrence of wake-up schedule 2: Not set setup

Network configuration (4- WWW ready)

WLAN settings: SWEETSHORTWPA edit
 Device name: YHUBWLN1-11D70 edit
 IP addressing: Automatic by DHCP edit
 (current IP: 192.168.1.54)

Incoming connections

Authentication to read information from the devices: NO edit
 Authentication to make changes to the devices: NO edit

Outgoing callbacks

Callback URL: http://www.stuckelberg.ch/testcb/doublebuff.php edit
 Callback method: POST Yocto-API
 Delay between callbacks: min: 1 [s] max: 30 [s]
 Network downtime to reboot: no downtime limit edit

Save Cancel

La fenêtre de configuration du YoctoHub-Wireless-SR

Ce contrôle d'accès est contrôlé depuis la section **Incoming connections**. Il peut se faire à deux niveaux distincts.

6.1. Accès "admin" protégé

Le mot de passe *admin* verrouille les accès en écriture sur les modules. Lorsqu'il est configuré, seuls les accès de type *admin* permettent d'accéder aux modules en lecture et en écriture. Les utilisateurs utilisant le login *admin* pourront éditer la configuration des modules vus par ce YoctoHub-Wireless-SR comme ils le souhaitent.

6.2. Accès "user" protégé

Le mot de passe *user* verrouille toute utilisation des modules. Lorsqu'il est configuré, toute utilisation sans mot de passe devient impossible. Les accès de type *user* ne permettent d'accéder aux modules qu'en lecture seule c'est-à-dire seulement pour consulter l'état des modules. Si vous instaurez simultanément un contrôle d'accès de type *user* et de type *admin*, les utilisateurs utilisant le login *user* ne pourront pas modifier la configuration des modules vus par ce YoctoHub-Wireless-SR.

Si vous configurez un accès *admin*, sans configurer d'accès *user*, les utilisateurs pourront continuer à consulter vos modules en lecture sans avoir à entrer de mot de passe.

Pour configurer l'accès au YoctoHub-Wireless-SR, cliquez sur le bouton **edit** des lignes **Authentication to read the information from the devices** ou **Authentication to write information to the devices**

6.3. Influence sur les API

Attention, le contrôle d'accès agira aussi sur les API Yoctopuce qui tenteront de se connecter à ce YoctoHub-Wireless-SR. Dans les API Yoctopuce, la gestion des droits d'accès est réalisée au niveau de l'appel à la fonction `RegisterHub()` : vous devrez donner l'adresse du YoctoHub-Wireless-SR sous la forme *login:password@adresse:port*, par exemple :

```
yRegisterHub("admin:mypass@192.168.0.10:4444",errmsg);
```

6.4. Effacement des mots de passe

Si vous perdez le mot passe de votre YoctoHub-Wireless-SR, vous pouvez reprendre le contrôle de votre module en réinitialisant tous ses réglages à la valeur par défaut. Pour ce faire, procurez-vous un câble USB pour le YoctoHub-Wireless-SR, et branchez-le à un ordinateur avec le VirtualHub¹ installé en maintenant le Yocto-bouton pressé. Ceci va forcer le YoctoHub-Wireless-SR à démarrer en mode de mise à jour du firmware. Il apparaît alors dans le VirtualHub en dessous de la liste des modules. Cliquez sur son numéro de série et choisissez un fichier de firmware à charger sur le module. Une fois le firmware rechargé avec cette méthode, le module sera réinitialisé avec les réglages d'usine, sans contrôle d'accès.

¹ <http://www.yoctopuce.com/FR/virtualhub.php>

7. Interactions avec l'extérieur

Le YoctoHub-Wireless-SR 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.

7.1. Configuration

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

Serial	Logical Name	Description	Action
YHUBWLN1-11D70		YoctoHub-Wireless	configure view log file beacon
MAXII001-121FD		Yocto-Maxi-IO	configure view log file beacon
RELAYHI1-00033		Yocto-PowerRelay	configure view log file beacon

Show device functions

Cliquez sur le bouton "Configure" de la première ligne

YHUBWLN1-11D70

Edit parameters for device YHUBWLN1-11D70, and click on the **Save** button.

Serial #: YHUBWLN1-11D70
 Product name: YoctoHub-Wireless
 Firmware: 12704 upgrade
 Logical name:
 Luminosity: (signal leds only)

Device functions

Each function of the device has a physical name and a logical name. You can change the logical name using the **rename** button.

YHUBWLN1-11D70.files / rename
 User files: 0 file, 3724 KB available manage files
 YHUBWLN1-11D70.hubPort1 / YWATMK1-0E24B rename
 YHUBWLN1-11D70.hubPort2 / rename
 YHUBWLN1-11D70.hubPort3 / RELAYHI-00033 rename
 YHUBWLN1-11D70.network / YHUBWLN1-11D70 rename
 YHUBWLN1-11D70.realTimeClock / rename
 YHUBWLN1-11D70.wakeUpMonitor / rename
 YHUBWLN1-11D70.wakeUpSchedule1 / rename
 YHUBWLN1-11D70.wakeUpSchedule2 / rename
 YHUBWLN1-11D70.wireless / rename

Wake-up Scheduler

Maximum power-on duration: no limit edit
 Next occurrence of wake-up schedule 1: Not set setup
 Next occurrence of wake-up schedule 2: Not set setup

Network configuration (4- WWW ready)

WLAN settings: SWEETSHORTWPA edit
 Device name: YHUBWLN1-11D70 edit
 IP addressing: Automatic by DHCP edit
 (current IP: 192.168.1.54)

Incoming connections

Authentication to read information from the devices: NO edit
 Authentication to make changes to the devices: NO edit

Outgoing callbacks

Callback URL: <http://www.stuckelberg.ch/testcb/doublebuff.php> edit
 Callback method: POST Yocto-API
 Delay between callbacks: min: 1 [s] max: 30 [s]
 Network downtime to reboot: no downtime limit edit

Save Cancel

Puis éditez la section *Outgoing callbacks*.

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

7.2. 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-Wireless-SR.

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 Yoctopuce¹.

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

7.3. 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 notamment pour la configuration à distance des modules Yoctopuce et la géolocalisation des mesures.

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

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

7.4. 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.

7.5. 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.

7.6. 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 Yoctopuce³.

Yoctopuce n'est en aucune manière affilié à PAESSLER.

7.7. 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.

7.8. 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-Wireless-SR sert alors de passerelle. Vous avez simplement à définir l'URL du script de contrôle sur le serveur HTTP et éventuellement les crédits nécessaires pour y accéder. Dans le script serveur, vous devez alors initialiser la librairie Yoctopuce avec l'appel suivant:

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

Il existe deux possibilités pour utiliser l'API Yoctopuce en mode callback. La première, disponible en PHP, Java et Node.JS est basée sur des connections en pur HTTP. Le YoctoHub-Wireless-SR poste l'intégralité de son état au serveur, et reçoit en retour des commandes du script sur le serveur. Ce

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>

fonctionnement implique quelques limitations: les interactions 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-Wireless-SR 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-Wireless-SR, même à travers un filtre NAT ou un pare-feu. Pour plus d'informations, référez-vous au blog de Yoctopuce ⁴.

7.9. User defined callback

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

Edit callback

This VirtualHub can post the advertised values of all devices on a specific URL on a regular basis. If you wish to use this feature, choose the callback type follow the steps below carefully.

1. Specify the Type of callback you want to use: **Yocto-API callback**
 Yoctopuce devices can be controlled through remote PHP scripts. That Yocto-API callback protocol is designed so it can pass through NAT filters without opening ports. See your device user manual, *PHP programming* section for more details.
2. Specify the URL to use for reporting values. *HTTPS protocol is not yet supported.*
 Callback URL:
3. If your callback requires authentication, enter credentials here. Digest authentication is recommended, but Basic authentication works as well.
 Username:
 Password:
4. Setup the desired frequency of notifications:
 No less than seconds between two notification
 But notify after seconds in any case
5. Press on the **Test** button to check your parameters.
6. When everything works, press on the **OK** button.

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-Wireless-SR 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-Wireless-SR poste avec la méthode POST les valeurs notifiées⁵ 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.

Tests

Afin de vous permettre de déboguer le processus, le YoctoHub-Wireless-SR 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**.

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-Wireless-SR.

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:

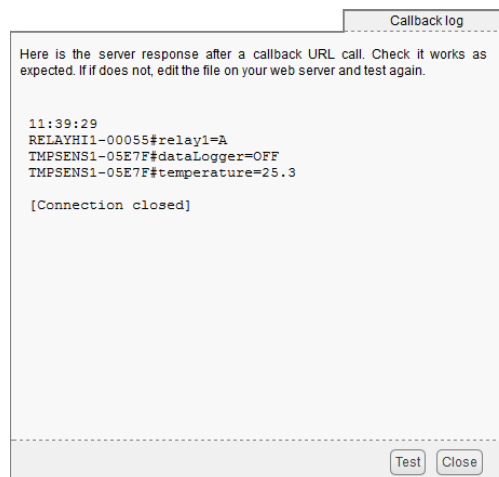
```
NOM_DU_MODULE#NOM_HARDWARE = VALUE
```

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.

8. Programmation

8.1. Accès aux modules connectés

Le YoctoHub-Wireless-SR 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-Wireless-SR se situe au niveau de l'appel à *registerHub*. Pour utiliser des modules USB connectés en natif, le paramètre de *RegisterHub* est *usb*, pour utiliser des modules connectés à un YoctoHub-Wireless-SR, il suffit de remplacer ce paramètre par l'adresse IP du YoctoHub-Wireless-SR. Par exemple, en Delphi:

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

devient

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

8.2. Contrôle du YoctoHub-Wireless-SR

Du point de vue API de programmation, le YoctoHub-Wireless-SR 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-Wireless-SR, etc.

Wireless

Cette classe permet de contrôler la configuration du réseau sans fil du YoctoHub-Wireless-SR, en particulier le SSID et la clé d'accès au réseau sans fil.

Network

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

HubPort

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

Files

Cette classe permet d'accéder aux fichiers stockés dans la mémoire flash du YoctoHub-Wireless-SR. Le YoctoHub-Wireless-SR 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-Wireless-SR.

WakeUpMonitor

Cette classe permet de contrôler la mise en sommeil du YoctoHub-Wireless-SR.

WakeUpSchedule

Cette classe permet d'agender un ou plusieurs réveils du YoctoHub-Wireless-SR.

Vous trouverez quelques exemples de contrôle du YoctoHub-Wireless-SR par programmation dans les bibliothèques Yoctopuce, disponibles gratuitement sur le site de Yoctopuce.

9. Mise en sommeil

Le YoctoHub-Wireless-SR dispose d'une horloge en temps 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.

9.1. Configuration manuelle du système de réveil

Les conditions de réveil peuvent être configurées manuellement en vous connectant sur l'interface du YoctoHub-Wireless-SR. Dans la section **Wake-up scheduler** de la fenêtre de configuration générale, cliquez sur le bouton setup correspondant à l'un des "wake-up-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 ignorées.

WakeUp schedule 1

Define wake up times: each button toggles a condition. A wake-up will occur when at least one condition per section is true. Sections without any condition defined are ignored.

Days in the week

Mon	Tue	Wed	Thu	Fri	Sat	Sun
-----	-----	-----	-----	-----	-----	-----

Days in the month

1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31					

set all every 2 every 3 clear

Months

Jan	Feb	Mar	Avr	May	Jun	Jui	Aug	Sep	Oct	Nov	Dec
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

set all every 2 every 3 clear

Hours

0	1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22	23

set all every 2 every 3 every 4 every 6 clear

Minutes

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55	56	57	58	59

set all every 2 every 3 every 5 every 10 clear

Ok Close

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-Wireless-SR le temps d'éveil maximal désiré, après lequel le module retournera automatiquement en sommeil profond. Si vous utilisez votre YoctoHub-Wireless-SR sur batteries, ceci vous assurera de ne pas vider les batteries même si aucun ordre de mise en sommeil explicite n'est reçu.

9.2. Paramétrage du système de réveil par logiciel

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

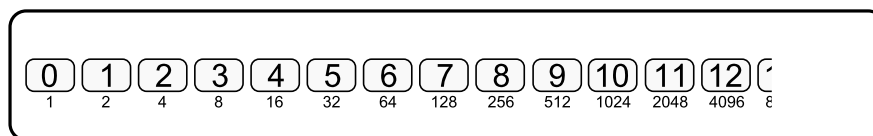
wakeUpMonitor

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-Wireless-SR peut rester éveil avant de se mettre automatiquement en sommeil.

wakeUpSchedule

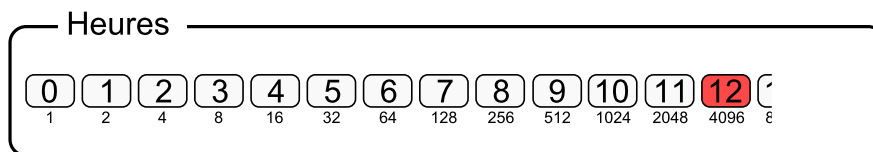
La fonction *wakeUpSchedule* permet de programmer une condition de réveil. Elle dispose de cinq variables qui permet de définir des correspondance sur les minutes, heure, jour de la semaine, jour dans le mois, et mois. Ces variables sont des variables entières dont chaque bit défini une correspondance.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

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.



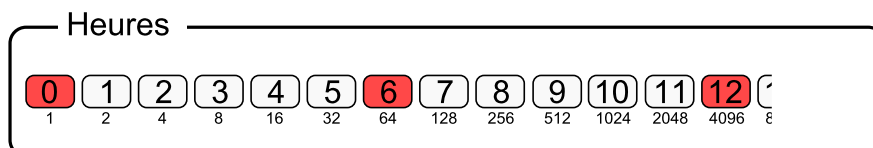
Chaque case se voit affecter une puissance de deux

Ainsi pour programmer le YoctoHub-Wireless-SR 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.

Heures

0	1	2	3	4	5	6	7	8	9	10	11	12
1	2	4	8	16	32	64	128	256	512	1024	2048	4096

$$64 + 4096 = 4060$$

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.

Jour de la semaine

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
1	2	4	8	16	32	64

2

Jour du mois

1	2	3	4	5	6	7	8	9	10	11	12	13	1
1	2	4	8	16	32	64	128	256	512	1024	2048	4096	81

$$1 + 2 + 4 + 8 + 16 + 32 + 64 = 127$$

Exemple de réveil tous les premiers mardi du mois

Certains langages 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 langages actuels.

MinutesA

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

MinutesB

30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55	56	57	58	59

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-Wireless-SR dispose de deux fonctions *wakeUpSchedule* ce qui permet de programmer jusqu'à deux types de réveils indépendants.

10. Personnalisation de l'interface Web

Votre YoctoHub-Wireless-SR dispose d'un petit système de fichiers embarqué, qui permet de stocker des fichiers personnalisés utilisables par le YoctoHub-Wireless-SR. Le système de fichiers se manipule grâce à la librairie *yocto_files*. Vous pourrez y stocker les fichiers de votre choix. Au besoin, vous pourrez y stocker une application Web permettant de gérer les modules connectés à votre YoctoHub-Wireless-SR.

10.1. Utilisation

Utilisation interactive

L'interface Web du YoctoHub-Wireless-SR fournit une interface sommaire pour manipuler le contenu du système de fichiers: cliquez simplement sur le bouton *configuration* correspondant à votre module dans l'interface du hub, puis sur le bouton *manage files*. Les fichiers présents sont listés, et vous pouvez les visualiser, les effacer ou en ajouter (téléchargement).

En raison de sa petite taille, le système de fichiers ne possède pas de notion explicite de répertoire. Vous pouvez toutefois utiliser la barre oblique "/" à l'intérieur des noms de fichiers pour les *classer* comme s'ils étaient dans des répertoires.

Utilisation programmée

Le système de fichiers s'utilise avec la librairie *yocto_files*. Les fonctions de bases sont disponibles:

- *upload* vous permet de créer un nouveau fichier sur le module, dont vous fournissez le contenu;
- *get_list* vous permet de connaître la liste de fichier présents sur le module, y compris la taille et le CRC32 du contenu;
- *download* vous permet de le récupérer dans une variable le contenu d'un fichier présent sur le module;
- *remove* permet d'effacer un fichier du module.
- *format* permet de réinitialiser le système de fichiers à un état vide, non fragmenté.

Un programme utilisant le système de fichier bien conçu devrait toujours commencer par s'assurer que les fichiers nécessaires à son fonctionnement sont présents sur le module, et si nécessaire les charger sur le module. Cela permet de gérer de manière transparente les mises à jour logicielles et le déploiement de l'application sur des nouveaux modules. Pour faciliter la détection des versions de fichiers présents sur le module, la méthode *get_list* retourne pour chaque fichier une signature sur 32 bit appelée CRC (Cyclic Redundancy Check), qui identifie de manière fiable le contenu du fichier. Ainsi, si le CRC du fichier correspond, il y a moins d'une chance sur 4 milliards que son contenu ne soit pas le bon. Vous pouvez même calculer dans votre programme par avance le CRC du contenu

que vous désirez, et ainsi le vérifier sans avoir à transférer le fichier. La fonction CRC utilisée par le système de fichiers Yoctopuce est la même que celle d'Ethernet, Gzip, PNG, etc. Sa valeur caractéristique pour la chaîne de neuf caractères "123456789" est 0xCBF43926.

Utilisation par HTTP

Les fichiers que vous avez chargés sur votre YoctoHub-Wireless-SR sont accessibles par HTTP, à la racine du module (au même niveau que l'API REST). Cela permet de charger par exemple des pages d'interface HTML et Javascript personnalisées. Vous ne pouvez toutefois pas remplacer le contenu d'un fichier préchargé sur le module, mais seulement en ajouter des nouveaux.

Interfaces utilisateur et optimisation

Puisque que pouvez sauvegarder des fichiers directement sur la mémoire flash du module et les accéder depuis l'extérieur, il est très facile de construire une application WEB pour contrôler les modules connectés au hub et de la stocker directement sur le hub. C'est un moyen très pratique pour construire des systèmes télécommandables depuis un smart-phone ou une tablette. Cependant le \$PRODANME\$ est plus limité qu'un serveur WEB normal: il n'accepte qu'un nombre limité de connexions en parallèle. La plupart des browsers WEB actuels ayant tendance à ouvrir un maximum de connexions en parallèle pour charger tous les éléments d'une page WEB, cela peut mener à des temps de chargement très long. Pour éviter cela, essayez de garder vos pages WEB aussi compactes que possible en incluant le code javascript et CSS directement dans la page. Si vous le pouvez, incluez aussi les images en base64.

10.2. Limitations

Le filesystem embarqué sur votre YoctoHub-Wireless-SR a quelques limitations techniques:

- Son espace de stockage maximal est 3.5Mo, répartis en blocs permettant de stocker jusqu'à environ 800 fichiers
- L'effacement d'un fichier ne récupère pas nécessairement immédiatement la totalité de la place utilisée par le fichier. L'espace non libéré sera entièrement réutilisé si l'on recrée un fichier du même nom, mais pas forcément si l'on crée des fichiers utilisant chaque fois des noms différents. Pour cette raison, il n'est pas recommandé de générer automatiquement des fichiers avec des noms toujours différents.
- L'espace non libéré peut être entièrement récupéré avec la commande *format*, qui libère la totalité des fichiers.
- Chaque mise à jour du firmware provoque implicitement un formatage complet du filesystem.
- Comme toutes les mémoires flash, la mémoire utilisée pour stocker les fichiers a une durée de vie de 100'000 cycles d'effacement environ. C'est assez, mais ce n'est pas illimité. Prenez donc garde à ne pas écrire et effacer inutilement des fichiers en boucle très rapidement, sous peine de détruire votre module.

11. Référence de l'API de haut niveau

Ce chapitre résume les fonctions de l'API de haut niveau pour commander votre YoctoHub-Wireless-SR. La syntaxe et les types précis peuvent varier d'un langage à l'autre mais, sauf avis contraire toutes sont disponibles dans chaque langage. 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-Wireless-SR.

11.1. La classe YHubPort

Interface pour interagir avec les ports de YoctoHub, disponibles par exemple dans le YoctoHub-Ethernet, le YoctoHub-GSM-4G, le YoctoHub-Shield et le YoctoHub-Wireless-n

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

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.FindHubPort()

YHubPort

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

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

YHubPort

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)
ts	static FindHubPortInContext (yctx : YAPIContext, func : string): YHubPort
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 où 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.FirstHubPort()

YHubPort

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

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

YHubPort

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

java	static YHubPort FirstHubPortInContext (YAPIContext yctx)
uwp	static YHubPort FirstHubPortInContext (YAPIContext yctx)
ts	static FirstHubPortInContext (yctx : YAPIContext): YHubPort null
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).

```
dnf static new string[] GetSimilarFunctions( )
```

```
cp static vector<string> GetSimilarFunctions( )
```

Chaque chaîne retournée peut être passée en argument à la méthode `YHubPort.FindHubPort` pour obtenir un objet permettant d'interagir 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.

`dnsp` `int Enabled`

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`.

dnf	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'identifiant matériel de la fonction (par exemple: `MyCustomName.relay1`)

hubport→**FunctionId****YHubPort**

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

dnf `string` **FunctionId**

Par exemple `re_lay1`.

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 exemple RELAYL01-123456 . r e l a y 1).

hubport→IsOnline**YHubPort**

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

dnf	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.

dnf **int** **PortState**

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 ()
ts	async clearCache (): Promise<void>
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 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 ()
ts	async describe (): Promise<string>
es	async describe ()

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès à 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 méthode va retourner `Relay(MyCustomName.relay1)=RELAYL01-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

Retourne :

une chaîne de caractères décrivant le port de YoctoHub (ex:
`Relay(MyCustomName.relay1)=RELAYL01-123456.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 ()
java	String get_advertisedValue ()
uwp	async Task<string> get_advertisedValue ()
py	get_advertisedValue ()
php	function get_advertisedValue ()
ts	async get_advertisedValue (): Promise<string>
es	async get_advertisedValue ()
dnp	string get_advertisedValue ()
cp	string 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 `YHubPort.AVERTISEDVALUE_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 ()
java	int get_baudRate ()
uwp	async Task<int> get_baudRate ()
py	get_baudRate ()
php	function get_baudRate ()
ts	async get_baudRate (): Promise<number>
es	async get_baudRate ()
dnp	int get_baudRate ()
cp	int 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 YHubPort . BAUDRATE_INVALID.

hubport→get_enabled()**YHubPort****hubport→enabled()**

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 ()
java	int get_enabled ()
uwp	async Task<int> get_enabled ()
py	get_enabled ()
php	function get_enabled ()
ts	async get_enabled (): Promise<YHubPort_Enabled>
es	async get_enabled ()
dnp	int get_enabled ()
cp	int get_enabled ()
cmd	YHubPort target get_enabled

Retourne :

soit `YHubPort.ENABLED_FALSE`, soit `YHubPort.ENABLED_TRUE`, selon vrai si le port du YoctoHub est alimenté, faux sinon

En cas d'erreur, déclenche une exception ou retourne `YHubPort.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 ()
ts	get_errorMessage (): string
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	<code>function get_errorType()</code>
cpp	<code>YRETCODE get_errorType()</code>
m	<code>-(YRETCODE) errorType</code>
pas	<code>YRETCODE get_errorType(): YRETCODE</code>
vb	<code>function get_errorType() As YRETCODE</code>
cs	<code>YRETCODE get_errorType()</code>
java	<code>int get_errorType()</code>
py	<code>get_errorType()</code>
php	<code>function get_errorType()</code>
ts	<code>get_errorType(): number</code>
es	<code>get_errorType()</code>

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 ()
java	String get_friendlyName ()
py	get_friendlyName ()
php	function get_friendlyName ()
ts	async get_friendlyName (): Promise<string>
es	async get_friendlyName ()
dnp	string get_friendlyName ()
cp	string 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'identifiant 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 `YHubPort.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 ()
ts	async get_functionDescriptor (): Promise<string>
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\$CLASSNAME\$.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 ()
java	String get_functionId ()
py	get_functionId ()
php	function get_functionId ()
ts	async get_functionId (): Promise<string>
es	async get_functionId ()
dnp	string get_functionId ()
cp	string get_functionId ()

Par exemple `relay1`.

Retourne :

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

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

hubport→get_hardwareId()**YHubPort****hubport→hardwareId()**

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 ()
java	String get_hardwareId ()
py	get_hardwareId ()
php	function get_hardwareId ()
ts	async get_hardwareId (): Promise<string>
es	async get_hardwareId ()
dnp	string get_hardwareId ()
cp	string get_hardwareId ()

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du port de YoctoHub (par exemple RELAYL01-123456 . relay1).

Retourne :

une chaîne de caractères identifiant le port de YoctoHub (ex: RELAYL01-123456 . relay1)

En cas d'erreur, déclenche une exception ou retourne YHubPort . 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 ()
java	String get_logicalName ()
uwp	async Task<string> get_logicalName ()
py	get_logicalName ()
php	function get_logicalName ()
ts	async get_logicalName (): Promise<string>
es	async get_logicalName ()
dnp	string get_logicalName ()
cp	string 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 YHubPort.LOGICALNAME_INVALID.

hubport→get_module()**YHubPort****hubport→module()**

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 ()
java	YModule get_module ()
py	get_module ()
php	function get_module ()
ts	async get_module (): Promise<YModule>
es	async get_module ()
dnp	YModuleProxy get_module ()
cp	YModuleProxy * 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()**YHubPort****hubport→portState()**

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 ()
java	int get_portState ()
uwp	async Task<int> get_portState ()
py	get_portState ()
php	function get_portState ()
ts	async get_portState (): Promise<YHubPort_PortState>
es	async get_portState ()
dnp	int get_portState ()
cp	int get_portState ()
cmd	YHubPort target get_portState

Retourne :

une valeur parmi YHubPort.PORTSTATE_OFF, YHubPort.PORTSTATE_OVRD, YHubPort.PORTSTATE_ON, YHubPort.PORTSTATE_RUN et YHubPort.PORTSTATE_PROG représentant l'état actuel du port de YoctoHub

En cas d'erreur, déclenche une exception ou retourne YHubPort.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 ()
java	String get_serialNumber ()
uwp	async Task<string> get_serialNumber ()
py	get_serialNumber ()
php	function get_serialNumber ()
ts	async get_serialNumber (): Promise<string>
es	async get_serialNumber ()
dnp	string get_serialNumber ()
cp	string 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 YFunction.SERIALNUMBER_INVALID.

hubport→get_userdata()**YHubPort****hubport→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 ()
ts	async get_userdata (): Promise<object null>
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 ()
java	boolean isOnline ()
py	isOnline ()
php	function isOnline ()
ts	async isOnline (): Promise<boolean>
es	async isOnline ()
dnp	bool isOnline ()
cp	bool 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 ()
java	boolean isReadOnly ()
uwp	async Task<bool> isReadOnly ()
py	isReadOnly ()
php	function isReadOnly ()
ts	async isReadOnly (): Promise<boolean>
es	async isReadOnly ()
dnp	bool isReadOnly ()
cp	bool 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)
ts	async load (msValidity : number): Promise<number>
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 occasionnellement 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(ByVal attrName As String) As String
cs	string loadAttribute(string attrName)
java	String loadAttribute(String attrName)
uwp	async Task<string> loadAttribute(string attrName)
py	loadAttribute(attrName)
php	function loadAttribute(\$attrName)
ts	async loadAttribute(attrName: string): Promise<string>
es	async loadAttribute(attrName)
dnp	string loadAttribute(string attrName)
cp	string loadAttribute(string 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 occasionnellement 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 ()
java	int muteValueCallbacks ()
uwp	async Task<int> muteValueCallbacks ()
py	muteValueCallbacks ()
php	function muteValueCallbacks ()
ts	async muteValueCallbacks (): Promise<number>
es	async muteValueCallbacks ()
dnp	int muteValueCallbacks ()
cp	int 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 déclenchement de callbacks HTTP. N'oubliez pas d'appeler la méthode `saveToF lash( )` 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	-(nullable 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 ()
ts	nextHubPort (): YHubPort null
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 _Nullable) callback
pas	LongInt registerValueCallback (callback : TYHubPortValueCallback): LongInt
vb	function registerValueCallback (ByVal callback As YHubPortValueCallback) 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)
ts	async registerValueCallback (callback : YHubPortValueCallback null): Promise<number>
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)
java	int set_enabled (int newval)
uwp	async Task<int> set_enabled (int newval)
py	set_enabled (newval)
php	function set_enabled (\$newval)
ts	async set_enabled (newval : YHubPort_Enabled): Promise<number>
es	async set_enabled (newval)
dnp	int set_enabled (int newval)
cp	int set_enabled (int 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 YHubPort.ENABLED_FALSE, soit YHubPort.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)
java	int set_logicalName (String newval)
uwp	async Task<int> set_logicalName (string newval)
py	set_logicalName (newval)
php	function set_logicalName (\$ newval)
ts	async set_logicalName (newval : string): Promise<number>
es	async set_logicalName (newval)
dnp	int set_logicalName (string newval)
cp	int set_logicalName (string 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)
ts	async set_userdata (data : object null): Promise<void>
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 ()
java	int unmuteValueCallbacks ()
uwp	async Task<int> unmuteValueCallbacks ()
py	unmuteValueCallbacks ()
php	function unmuteValueCallbacks ()
ts	async unmuteValueCallbacks (): Promise<number>
es	async unmuteValueCallbacks ()
dnp	int unmuteValueCallbacks ()
cp	int 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	<code>function wait_async(callback, context)</code>
ts	<code>wait_async(callback: Function, context: object)</code>
es	<code>wait_async(callback, context)</code>

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.

11.2. La classe YWireless

Interface pour interagir avec les interfaces réseau sans fil, disponibles par exemple dans le YoctoHub-Wireless, le YoctoHub-Wireless-SR, le YoctoHub-Wireless-g et le YoctoHub-Wireless-n

La classe `YWireless` permet de configurer et de contrôler la configuration du réseau sans fil 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	<code><script type='text/javascript' src='yocto_wireless.js'></script></code>
cpp	<code>#include "yocto_wireless.h"</code>
m	<code>#import "yocto_wireless.h"</code>
pas	<code>uses yocto_wireless;</code>
vb	<code>yocto_wireless.vb</code>
cs	<code>yocto_wireless.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YWireless;</code>
uwp	<code>import com.yoctopuce.YoctoAPI.YWireless;</code>
py	<code>from yocto_wireless import *</code>
php	<code>require_once('yocto_wireless.php');</code>
ts	<code>in HTML: import { YWireless } from '../dist/esm/yocto_wireless.js'; in Node.js: import { YWireless } from 'yoctolib-cjs/yocto_wireless.js';</code>
es	<code>in HTML: <script src="../lib/yocto_wireless.js"></script> in node.js: require('yoctolib-es2017/yocto_wireless.js');</code>
dnf	<code>import YoctoProxyAPI.YWirelessProxy</code>
cp	<code>#include "yocto_wireless_proxy.h"</code>
vi	<code>YWireless.vi</code>
ml	<code>import YoctoProxyAPI.YWirelessProxy</code>

Fonction globales

YWireless.FindWireless(func)

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

YWireless.FindWirelessInContext(yctx, func)

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

YWireless.FirstWireless()

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

YWireless.FirstWirelessInContext(yctx)

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

YWireless.GetSimilarFunctions()

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

Propriétés des objets YWirelessProxy

wireless→AdvertisedValue [lecture seule]

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

wireless→FriendlyName [lecture seule]

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

wireless→FunctionId [lecture seule]

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

wireless→HardwareId [lecture seule]

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

wireless→IsOnline [lecture seule]

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

wireless→LinkQuality [lecture seule]

Qualité de la connection, exprimée en pourcents.

wireless→LogicalName [modifiable]

Nom logique de la fonction.

wireless→SerialNumber [lecture seule]

Numéro de série du module, préprogrammé en usine.

wireless→Ssid [lecture seule]

Nom (SSID) du réseau sans fil sélectionné.

Méthodes des objets YWireless

wireless→adhocNetwork(ssid, securityKey)

Modifie la configuration de l'interface réseau sans fil pour créer un réseau sans fil sans point d'accès, en mode "ad-hoc".

wireless→clearCache()

Invalide le cache.

wireless→describe()

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

wireless→get_advertisedValue()

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

wireless→get_channel()

Retourne le numéro du canal 802.11 utilisé, ou 0 si le réseau sélectionné n'a pas été trouvé.

wireless→get_detectedWlans()

Retourne une liste d'objets objet YWlanRecord qui décrivent les réseaux sans fils détectés.

wireless→get_errorMessage()

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

wireless→get_errorType()

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

wireless→get_friendlyName()

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

wireless→get_functionDescriptor()

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

wireless→get_functionId()

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

wireless→get_hardwareId()

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

wireless→get_linkQuality()

Retourne la qualité de la connection, exprimée en pourcents.

wireless→get_logicalName()

Retourne le nom logique de l'interface réseau sans fil.

wireless→get_message()

	Retourne le dernier message de diagnostic de l'interface au réseau sans fil.
wireless→get_module()	Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.
wireless→get_module_async(callback, context)	Retourne l'objet YModule correspondant au module Yoctopuce qui héberge la fonction.
wireless→get_security()	Retourne l'algorithme de sécurité utilisé par le réseau sans fil sélectionné.
wireless→get_serialNumber()	Retourne le numéro de série du module, préprogrammé en usine.
wireless→get_ssid()	Retourne le nom (SSID) du réseau sans fil sélectionné.
wireless→get_userData()	Retourne le contenu de l'attribut userData, précédemment stocké à l'aide de la méthode set_userData.
wireless→get_wlanState()	Retourne l'état actuel de l'interface réseau sans fil.
wireless→isOnline()	Vérifie si le module hébergeant l'interface réseau sans fil est joignable, sans déclencher d'erreur.
wireless→isOnline_async(callback, context)	Vérifie si le module hébergeant l'interface réseau sans fil est joignable, sans déclencher d'erreur.
wireless→isReadOnly()	Test si la fonction est en lecture seule.
wireless→joinNetwork(ssid, securityKey)	Modifie la configuration de l'interface réseau sans fil pour se connecter à un point d'accès sans fil existant (mode "infrastructure").
wireless→load(msValidity)	Met en cache les valeurs courantes de l'interface réseau sans fil, avec une durée de validité spécifiée.
wireless→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.
wireless→load_async(msValidity, callback, context)	Met en cache les valeurs courantes de l'interface réseau sans fil, avec une durée de validité spécifiée.
wireless→muteValueCallbacks()	Désactive l'envoi de chaque changement de la valeur publiée au hub parent.
wireless→nextWireless()	Continue l'énumération des interfaces réseau sans fil commencée à l'aide de yFirstWireless() Attention, vous ne pouvez faire aucune supposition sur l'ordre dans lequel les interfaces réseau sans fil sont retournés.
wireless→registerValueCallback(callback)	Enregistre la fonction de callback qui est appelée à chaque changement de la valeur publiée.
wireless→set_logicalName(newval)	Modifie le nom logique de l'interface réseau sans fil.
wireless→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.
wireless→softAPNetwork(ssid, securityKey)	Modifie la configuration de l'interface réseau sans fil pour créer un pseudo point d'accès sans fil ("Soft AP").

wireless→startWlanScan()

Déclenche un balayage des fréquences utilisable et construit la liste de réseau sans fils disponible.

wireless→unmuteValueCallbacks()

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

wireless→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.

YWireless.FindWireless()

YWireless.FindWireless()

YWireless

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

js	function yFindWireless (func)
cpp	YWireless* FindWireless (string func)
m	+(YWireless*) FindWireless : (NSString*) func
pas	TYWireless yFindWireless (func : string): TYWireless
vb	function FindWireless (ByVal func As String) As YWireless
cs	static YWireless FindWireless (string func)
java	static YWireless FindWireless (String func)
uwp	static YWireless FindWireless (string func)
py	FindWireless (func)
php	function FindWireless (\$func)
ts	static FindWireless (func : string): YWireless
es	static FindWireless (func)
dnp	static YWirelessProxy FindWireless (string func)
cp	static YWirelessProxy * FindWireless (string 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 sans fil soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YWireless.isOnline()` pour tester si l'interface réseau sans fil 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 l'application.

Paramètres :

func une chaîne de caractères qui référence l'interface réseau sans fil sans ambiguïté, par exemple `YHUBWLN1.wireless`.

Retourne :

un objet de classe `YWireless` qui permet ensuite de contrôler l'interface réseau sans fil.

YWireless.FindWirelessInContext()

YWireless.FindWirelessInContext()

YWireless

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

```

java static YWireless FindWirelessInContext( YAPIContext yctx, String func)
uwp static YWireless FindWirelessInContext( YAPIContext yctx, string func)
ts static FindWirelessInContext( yctx: YAPIContext, func: string): YWireless
es static FindWirelessInContext( 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 sans fil soit en ligne au moment où elle est appelée, l'objet retourné sera néanmoins valide. Utiliser la méthode `YWireless.isOnline()` pour tester si l'interface réseau sans fil 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 fil sans ambiguïté, par exemple `YHUBWLN1.wireless`.

Retourne :

un objet de classe `YWireless` qui permet ensuite de contrôler l'interface réseau sans fil.

YWireless.FirstWireless()

YWireless.FirstWireless()

YWireless

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

js	function yFirstWireless ()
cpp	YWireless * FirstWireless ()
m	+(YWireless*) FirstWireless
pas	TYWireless yFirstWireless (): TYWireless
vb	function FirstWireless () As YWireless
cs	static YWireless FirstWireless ()
java	static YWireless FirstWireless ()
uwp	static YWireless FirstWireless ()
py	FirstWireless ()
php	function FirstWireless ()
ts	static FirstWireless (): YWireless null
es	static FirstWireless ()

Utiliser la fonction `YWireless.nextWireless()` pour itérer sur les autres interfaces réseau sans fil.

Retourne :

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

YWireless.FirstWirelessInContext()

YWireless.FirstWirelessInContext()

YWireless

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

java	static YWireless FirstWirelessInContext (YAPIContext yctx)
uwp	static YWireless FirstWirelessInContext (YAPIContext yctx)
ts	static FirstWirelessInContext (yctx : YAPIContext): YWireless null
es	static FirstWirelessInContext (yctx)

Utiliser la fonction `YWireless.nextWireless()` pour itérer sur les autres interfaces réseau sans fil.

Paramètres :

yctx un contexte YAPI.

Retourne :

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

YWireless.GetSimilarFunctions()**YWireless****YWireless.GetSimilarFunctions()**

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

dnp	<code>static new string[] GetSimilarFunctions()</code>
-----	---

cp	<code>static vector<string> GetSimilarFunctions()</code>
----	---

Chaque chaîne retournée peut être passée en argument à la méthode `YWireless.FindWireless` pour obtenir un objet permettant d'interagir avec le module correspondant.

Retourne :

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

wireless→**AdvertisedValue****YWireless**

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

dnp

 string **AdvertisedValue**

wireless→FriendlyName**YWireless**

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

dnf	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'identifiant matériel de la fonction (par exemple: `MyCustomName.relay1`)

wireless→FunctionId**YWireless**

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

dnf `string FunctionId`

Par exemple `re_lay1`.

wireless→HardwareId**YWireless**

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 exemple RELAYL01-123456 . r e l a y 1).

wireless→IsOnline**YWireless**

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

`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.

wireless→LinkQuality

YWireless

Qualité de la connexion, exprimée en pourcents.

dnp `int LinkQuality`

wireless→LogicalName**YWireless**

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.

wireless→SerialNumber

YWireless

Numéro de série du module, préprogrammé en usine.

`dnp` `string` **SerialNumber**

wireless→Ssid**YWireless**

Nom (SSID) du réseau sans fil sélectionné.

dnp `string Ssid`

wireless→adhocNetwork()**YWireless**

Modifie la configuration de l'interface réseau sans fil pour créer un réseau sans fil sans point d'accès, en mode "ad-hoc".

js	function adhocNetwork (ssid , securityKey)
cpp	int adhocNetwork (string ssid , string securityKey)
m	-(int) adhocNetwork : (NSString*) ssid : (NSString*) securityKey
pas	LongInt adhocNetwork (ssid : string, securityKey : string): LongInt
vb	function adhocNetwork (ByVal ssid As String, ByVal securityKey As String) As Integer
cs	int adhocNetwork (string ssid , string securityKey)
java	int adhocNetwork (String ssid , String securityKey)
uwp	async Task<int> adhocNetwork (string ssid , string securityKey)
py	adhocNetwork (ssid , securityKey)
php	function adhocNetwork (\$ssid, \$securityKey)
ts	async adhocNetwork (ssid : string, securityKey : string): Promise<number>
es	async adhocNetwork (ssid , securityKey)
dnp	int adhocNetwork (string ssid , string securityKey)
cp	int adhocNetwork (string ssid , string securityKey)
cmd	YWireless target adhocNetwork ssid securityKey

Sur le YoctoHub-Wireless-g et le YoctoHub-Wireless-n, il faut utiliser à la place la fonction `softAPNetwork()` qui crée un pseudo point d'accès, plus efficace et mieux supporté qu'un réseau ad-hoc.

Si une clef d'accès est configurée pour un réseau ad-hoc, le réseau sera protégé par une sécurité WEP40 (5 caractères ou 10 chiffres hexadécimaux) ou WEP128 (13 caractères ou 26 chiffres hexadécimaux). Pour réduire les risques d'intrusion, il est recommandé d'utiliser une clé WEP128 basée sur 26 chiffres hexadécimaux provenant d'une bonne source aléatoire.

N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

ssid nom du réseau sans fil à créer
securityKey clé d'accès de réseau, 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.

wireless→clearCache()**YWireless**

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 ()
ts	async clearCache (): Promise<void>
es	async clearCache ()

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

wireless→describe()**YWireless**

Retourne un court texte décrivant de manière non-ambigüe l'instance de l'interface réseau sans fil 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 ()
ts	async describe (): Promise<string>
es	async describe ()

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès à 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 méthode va retourner Relay(MyCustomName.relay1)=RELAYL01-123456.relay1 si le module est déjà connecté ou Relay(BadCustomName.relay1)=unresolved si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

Retourne :

une chaîne de caractères décrivant l'interface réseau sans fil (ex:
Relay(MyCustomName.relay1)=RELAYL01-123456.relay1)

wireless→get_advertisedValue()**YWireless****wireless→advertisedValue()**

Retourne la valeur courante de l'interface réseau sans fil (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 ()
java	String get_advertisedValue ()
uwp	async Task<string> get_advertisedValue ()
py	get_advertisedValue ()
php	function get_advertisedValue ()
ts	async get_advertisedValue (): Promise<string>
es	async get_advertisedValue ()
dnp	string get_advertisedValue ()
cp	string get_advertisedValue ()
cmd	YWireless target get_advertisedValue

Retourne :

une chaîne de caractères représentant la valeur courante de l'interface réseau sans fil (pas plus de 6 caractères).

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

wireless→get_channel()**YWireless****wireless→channel()**

Retourne le numéro du canal 802.11 utilisé, ou 0 si le réseau sélectionné n'a pas été trouvé.

js	function get_channel ()
cpp	int get_channel ()
m	-(int) channel
pas	LongInt get_channel (): LongInt
vb	function get_channel () As Integer
cs	int get_channel ()
java	int get_channel ()
uwp	async Task<int> get_channel ()
py	get_channel ()
php	function get_channel ()
ts	async get_channel (): Promise<number>
es	async get_channel ()
dnp	int get_channel ()
cp	int get_channel ()
cmd	YWireless target get_channel

Retourne :

un entier représentant le numéro du canal 802.11 utilisé, ou 0 si le réseau sélectionné n'a pas été trouvé

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

wireless→get_detectedWlans()**YWireless****wireless→detectedWlans()**

Retourne une liste d'objets objet `YWlanRecord` qui décrivent les réseaux sans fils détectés.

js	function get_detectedWlans ()
cpp	vector<YWlanRecord> get_detectedWlans ()
m	-(NSMutableArray*) detectedWlans
pas	TYWlanRecordArray get_detectedWlans (): TYWlanRecordArray
vb	function get_detectedWlans () As List
cs	List<YWlanRecord> get_detectedWlans ()
java	ArrayList<YWlanRecord> get_detectedWlans ()
uwp	async Task<List<YWlanRecord>> get_detectedWlans ()
py	get_detectedWlans ()
php	function get_detectedWlans ()
ts	async get_detectedWlans (): Promise<YWlanRecord[]
es	async get_detectedWlans ()
dnp	YWlanRecordProxy[] get_detectedWlans ()
cp	vector<YWlanRecordProxy> get_detectedWlans ()
cmd	YWireless target get_detectedWlans

La liste n'est pas mise à jour quand le module est déjà connecté à un accès sans fil (mode "infrastructure"). Pour forcer la détection des réseaux sans fil, il faut appeler `startWlanScan()`. L'appelant est responsable de la désallocation de la liste retournée dans les langages ne disposant pas de "garbage collection".

Retourne :

une liste d'objets `YWlanRecord`, contenant le SSID, le canal, la qualité du signal, et l'algorithme de sécurité utilisé par le réseau sans fil

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

wireless→get_errorMessage()**YWireless****wireless→errorMessage()**

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

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 ()
ts	get_errorMessage (): string
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 sans fil.

wireless→get_errorType()**YWireless****wireless→errorType()**

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

js	<code>function get_errorType()</code>
cpp	<code>YRETCODE get_errorType()</code>
m	<code>-(YRETCODE) errorType</code>
pas	<code>YRETCODE get_errorType(): YRETCODE</code>
vb	<code>function get_errorType() As YRETCODE</code>
cs	<code>YRETCODE get_errorType()</code>
java	<code>int get_errorType()</code>
py	<code>get_errorType()</code>
php	<code>function get_errorType()</code>
ts	<code>get_errorType(): number</code>
es	<code>get_errorType()</code>

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 sans fil.

wireless→get_friendlyName()**YWireless****wireless→friendlyName()**

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

js	function get_friendlyName ()
cpp	string get_friendlyName ()
m	-(NSString*) friendlyName
cs	string get_friendlyName ()
java	String get_friendlyName ()
py	get_friendlyName ()
php	function get_friendlyName ()
ts	async get_friendlyName (): Promise<string>
es	async get_friendlyName ()
dnp	string get_friendlyName ()
cp	string get_friendlyName ()

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

Retourne :

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

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

wireless→get_functionDescriptor()**YWireless****wireless→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 ()
ts	async get_functionDescriptor (): Promise<string>
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\$CLASSNAME\$.FUNCTIONDESCRIPTOR_INVALID

wireless→get_functionId()**YWireless****wireless→functionId()**

Retourne l'identifiant matériel de l'interface réseau sans fil, 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 ()
java	String get_functionId ()
py	get_functionId ()
php	function get_functionId ()
ts	async get_functionId (): Promise<string>
es	async get_functionId ()
dnp	string get_functionId ()
cp	string get_functionId ()

Par exemple `relay1`.

Retourne :

une chaîne de caractères identifiant l'interface réseau sans fil (ex: `relay1`)

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

wireless→get_hardwareId()**YWireless****wireless→hardwareId()**

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

js	function <code>get_hardwareId()</code>
cpp	string <code>get_hardwareId()</code>
m	-(NSString*) <code>hardwareId</code>
vb	function <code>get_hardwareId()</code> As String
cs	string <code>get_hardwareId()</code>
java	String <code>get_hardwareId()</code>
py	<code>get_hardwareId()</code>
php	function <code>get_hardwareId()</code>
ts	async <code>get_hardwareId()</code> : Promise<string>
es	async <code>get_hardwareId()</code>
dnp	string <code>get_hardwareId()</code>
cp	string <code>get_hardwareId()</code>

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'interface réseau sans fil (par exemple `RELAYL01-123456.relay1`).

Retourne :

une chaîne de caractères identifiant l'interface réseau sans fil (ex: `RELAYL01-123456.relay1`)

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

wireless→get_linkQuality()**YWireless****wireless→linkQuality()**

Retourne la qualité de la connexion, 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 ()
java	int get_linkQuality ()
uwp	async Task<int> get_linkQuality ()
py	get_linkQuality ()
php	function get_linkQuality ()
ts	async get_linkQuality (): Promise<number>
es	async get_linkQuality ()
dnp	int get_linkQuality ()
cp	int get_linkQuality ()
cmd	YWireless target get_linkQuality

Retourne :

un entier représentant la qualité de la connexion, exprimée en pourcents

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

wireless→get_logicalName()**YWireless****wireless→logicalName()**

Retourne le nom logique de l'interface réseau sans fil.

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 ()
java	String get_logicalName ()
uwp	async Task<string> get_logicalName ()
py	get_logicalName ()
php	function get_logicalName ()
ts	async get_logicalName (): Promise<string>
es	async get_logicalName ()
dnp	string get_logicalName ()
cp	string get_logicalName ()
cmd	YWireless target get_logicalName

Retourne :

une chaîne de caractères représentant le nom logique de l'interface réseau sans fil.

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

wireless→get_message()**wireless→message()**

Retourne le dernier message de diagnostic 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 ()
java	String get_message ()
uwp	async Task<string> get_message ()
py	get_message ()
php	function get_message ()
ts	async get_message (): Promise<string>
es	async get_message ()
dnp	string get_message ()
cp	string get_message ()
cmd	YWireless target get_message

Retourne :

une chaîne de caractères représentant le dernier message de diagnostic de l'interface au réseau sans fil

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

wireless→get_module()**YWireless****wireless→module()**

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 ()
java	YModule get_module ()
py	get_module ()
php	function get_module ()
ts	async get_module (): Promise<YModule>
es	async get_module ()
dnp	YModuleProxy get_module ()
cp	YModuleProxy * 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

wireless→get_module_async()**YWireless****wireless→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.

wireless→get_security()**YWireless****wireless→security()**

Retourne l'algorithme de sécurité utilisé par le réseau sans fil sélectionné.

js	function get_security ()
cpp	Y_SECURITY_enum get_security ()
m	-(Y_SECURITY_enum) security
pas	Integer get_security (): Integer
vb	function get_security () As Integer
cs	int get_security ()
java	int get_security ()
uwp	async Task<int> get_security ()
py	get_security ()
php	function get_security ()
ts	async get_security (): Promise<YWireless_Security>
es	async get_security ()
dnp	int get_security ()
cp	int get_security ()
cmd	YWireless target get_security

Retourne :

une valeur parmi YWireless.SECURITY_UNKNOWN, YWireless.SECURITY_OPEN, YWireless.SECURITY_WEP, YWireless.SECURITY_WPA et YWireless.SECURITY_WPA2 représentant l'algorithme de sécurité utilisé par le réseau sans fil sélectionné

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

wireless→get_serialNumber()**YWireless****wireless→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 ()
java	String get_serialNumber ()
uwp	async Task<string> get_serialNumber ()
py	get_serialNumber ()
php	function get_serialNumber ()
ts	async get_serialNumber (): Promise<string>
es	async get_serialNumber ()
dnp	string get_serialNumber ()
cp	string get_serialNumber ()
cmd	YWireless 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 YFunction.SERIALNUMBER_INVALID.

wireless→get_ssid()**YWireless****wireless→ssid()**

Retourne le nom (SSID) du réseau sans fil sélectionné.

js	function get_ssid ()
cpp	string get_ssid ()
m	-(NSString*) ssid
pas	string get_ssid (): string
vb	function get_ssid () As String
cs	string get_ssid ()
java	String get_ssid ()
uwp	async Task<string> get_ssid ()
py	get_ssid ()
php	function get_ssid ()
ts	async get_ssid (): Promise<string>
es	async get_ssid ()
dnp	string get_ssid ()
cp	string get_ssid ()
cmd	YWireless target get_ssid

Retourne :

une chaîne de caractères représentant le nom (SSID) du réseau sans fil sélectionné

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

wireless→get_userdata()**YWireless****wireless→userdata()**

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

js	<code>function get_userdata()</code>
cpp	<code>void * get_userdata()</code>
m	<code>-(id) userData</code>
pas	<code>Tobject get_userdata(): Tobject</code>
vb	<code>function get_userdata() As Object</code>
cs	<code>object get_userdata()</code>
java	<code>Object get_userdata()</code>
py	<code>get_userdata()</code>
php	<code>function get_userdata()</code>
ts	<code>async get_userdata(): Promise<object null></code>
es	<code>async get_userdata()</code>

Cet attribut n'est 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.

wireless→get_wlanState()**YWireless****wireless→wlanState()**

Retourne l'état actuel de l'interface réseau sans fil.

js	function get_wlanState ()
cpp	Y_WLANSTATE_enum get_wlanState ()
m	-(Y_WLANSTATE_enum) wlanState
pas	Integer get_wlanState (): Integer
vb	function get_wlanState () As Integer
cs	int get_wlanState ()
java	int get_wlanState ()
uwp	async Task<int> get_wlanState ()
py	get_wlanState ()
php	function get_wlanState ()
ts	async get_wlanState (): Promise<YWireless_WlanState>
es	async get_wlanState ()
dnp	int get_wlanState ()
cp	int get_wlanState ()
cmd	YWireless target get_wlanState

L'état `YWireless.WLANSTATE_DOWN` indique que l'interface n'est connectée à aucun réseau. L'état `YWireless.WLANSTATE_SCANNING` signifie que la carte réseau effectue un balayage des fréquences utilisables. Dans cet état le module n'est pas joignable et la configuration réseau n'est pas encore appliquée. L'état `YWireless.WLANSTATE_CONNECTED` signifie que la configuration réseau a pu être appliquée et que le module est joignable. Si l'interface est configurée pour fonctionner en mode ad-hoc ou Soft AP, cela signifie que le réseau est fonctionnel et que un périphérique peut se connecter au réseau. L'état `YWireless.WLANSTATE_REJECTED` signifie que l'interface réseau n'a pas pu joindre le réseau souhaité. la source de l'erreur peut être obtenue à l'aide de la méthode `get_message()`.

Retourne :

une valeur parmi `YWireless.WLANSTATE_DOWN`, `YWireless.WLANSTATE_SCANNING`, `YWireless.WLANSTATE_CONNECTED` et `YWireless.WLANSTATE_REJECTED` représentant l'état actuel de l'interface réseau sans fil

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

wireless→isOnline()**YWireless**

Vérifie si le module hébergeant l'interface réseau sans fil 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 ()
java	boolean isOnline ()
py	isOnline ()
php	function isOnline ()
ts	async isOnline (): Promise<boolean>
es	async isOnline ()
dnp	bool isOnline ()
cp	bool isOnline ()

Si les valeurs des attributs en cache de l'interface réseau sans fil 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 sans fil est joignable, `false` sinon

wireless→isOnline_async()**YWireless**

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

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

Si les valeurs des attributs en cache de l'interface réseau sans fil 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.

wireless→isReadOnly()

YWireless

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 ()
java	boolean isReadOnly ()
uwp	async Task<bool> isReadOnly ()
py	isReadOnly ()
php	function isReadOnly ()
ts	async isReadOnly (): Promise<boolean>
es	async isReadOnly ()
dnp	bool isReadOnly ()
cp	bool isReadOnly ()
cmd	YWireless 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

wireless→joinNetwork()**YWireless**

Modifie la configuration de l'interface réseau sans fil pour se connecter à un point d'accès sans fil existant (mode "infrastructure").

js	function joinNetwork (ssid , securityKey)
cpp	int joinNetwork (string ssid , string securityKey)
m	-(int) joinNetwork : (NSString*) ssid : (NSString*) securityKey
pas	LongInt joinNetwork (ssid : string, securityKey : string): LongInt
vb	function joinNetwork (ByVal ssid As String, ByVal securityKey As String) As Integer
cs	int joinNetwork (string ssid , string securityKey)
java	int joinNetwork (String ssid , String securityKey)
uwp	async Task<int> joinNetwork (string ssid , string securityKey)
py	joinNetwork (ssid , securityKey)
php	function joinNetwork (\$ssid, \$securityKey)
ts	async joinNetwork (ssid : string, securityKey : string): Promise<number>
es	async joinNetwork (ssid , securityKey)
dnp	int joinNetwork (string ssid , string securityKey)
cp	int joinNetwork (string ssid , string securityKey)
cmd	YWireless target joinNetwork ssid securityKey

N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

- ssid** nom du réseau sans fil à utiliser
- securityKey** clé d'accès au réseau, 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.

wireless→load()**YWireless**

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

js	<code>function load(msValidity)</code>
cpp	<code>YRETCODE load(int msValidity)</code>
m	<code>-(YRETCODE) load : (u64) msValidity</code>
pas	<code>YRETCODE load(msValidity: u64): YRETCODE</code>
vb	<code>function load(ByVal msValidity As Long) As YRETCODE</code>
cs	<code>YRETCODE load(ulong msValidity)</code>
java	<code>int load(long msValidity)</code>
py	<code>load(msValidity)</code>
php	<code>function load(\$msValidity)</code>
ts	<code>async load(msValidity: number): Promise<number></code>
es	<code>async load(msValidity)</code>

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 occasionnellement 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.

wireless→loadAttribute()**YWireless**

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(ByVal attrName As String) As String
cs	string loadAttribute(string attrName)
java	String loadAttribute(String attrName)
uwp	async Task<string> loadAttribute(string attrName)
py	loadAttribute(attrName)
php	function loadAttribute(\$attrName)
ts	async loadAttribute(attrName: string): Promise<string>
es	async loadAttribute(attrName)
dnp	string loadAttribute(string attrName)
cp	string loadAttribute(string 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.

wireless→load_async()**YWireless**

Met en cache les valeurs courantes de l'interface réseau sans fil, 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 occasionnellement 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-quiel à la fonction de callback

Retourne :

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

wireless→muteValueCallbacks()**YWireless**

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 ()
java	int muteValueCallbacks ()
uwp	async Task<int> muteValueCallbacks ()
py	muteValueCallbacks ()
php	function muteValueCallbacks ()
ts	async muteValueCallbacks (): Promise<number>
es	async muteValueCallbacks ()
dnp	int muteValueCallbacks ()
cp	int muteValueCallbacks ()
cmd	YWireless 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 déclenchement 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.

wireless→nextWireless()**YWireless**

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

js	function nextWireless ()
cpp	YWireless * nextWireless ()
m	-(nullable YWireless*) nextWireless
pas	TYWireless nextWireless (): TYWireless
vb	function nextWireless () As YWireless
cs	YWireless nextWireless ()
java	YWireless nextWireless ()
uwp	YWireless nextWireless ()
py	nextWireless ()
php	function nextWireless ()
ts	nextWireless (): YWireless null
es	nextWireless ()

Si vous souhaitez retrouver une interface réseau sans fil spécifique, utilisez `Wireless.findWireless()` avec un `hardwareID` ou un nom logique.

Retourne :

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

wireless→registerValueCallback()**YWireless**

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

js	function registerValueCallback (callback)
cpp	int registerValueCallback (YWirelessValueCallback callback)
m	-(int) registerValueCallback : (YWirelessValueCallback _Nullable) callback
pas	LongInt registerValueCallback (callback : TYWirelessValueCallback): LongInt
vb	function registerValueCallback (ByVal callback As YWirelessValueCallback) 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)
ts	async registerValueCallback (callback : YWirelessValueCallback null): Promise<number>
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'objet fonction dont la valeur a changé, et la chaîne de caractère décrivant la nouvelle valeur publiée.

wireless→set_logicalName()**YWireless****wireless→setLogicalName()**

Modifie le nom logique de l'interface réseau sans fil.

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)
java	int set_logicalName (String newval)
uwp	async Task<int> set_logicalName (string newval)
py	set_logicalName (newval)
php	function set_logicalName (\$newval)
ts	async set_logicalName (newval : string): Promise<number>
es	async set_logicalName (newval)
dnp	int set_logicalName (string newval)
cp	int set_logicalName (string newval)
cmd	YWireless 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 sans fil.

Retourne :

YAPI . SUCCESS si l'appel se déroule sans erreur.

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

wireless→set_userdata()**YWireless****wireless→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)
ts	async set_userdata (data : object null): Promise<void>
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

wireless→softAPNetwork()**YWireless**

Modifie la configuration de l'interface réseau sans fil pour créer un pseudo point d'accès sans fil ("Soft AP").

js	<code>function softAPNetwork(ssid, securityKey)</code>
cpp	<code>int softAPNetwork(string ssid, string securityKey)</code>
m	<code>-(int) softAPNetwork : (NSString*) ssid : (NSString*) securityKey</code>
pas	<code>LongInt softAPNetwork(ssid: string, securityKey: string): LongInt</code>
vb	<code>function softAPNetwork(ByVal ssid As String, ByVal securityKey As String) As Integer</code>
cs	<code>int softAPNetwork(string ssid, string securityKey)</code>
java	<code>int softAPNetwork(String ssid, String securityKey)</code>
uwp	<code>async Task<int> softAPNetwork(string ssid, string securityKey)</code>
py	<code>softAPNetwork(ssid, securityKey)</code>
php	<code>function softAPNetwork(\$ssid, \$securityKey)</code>
ts	<code>async softAPNetwork(ssid: string, securityKey: string): Promise<number></code>
es	<code>async softAPNetwork(ssid, securityKey)</code>
dnp	<code>int softAPNetwork(string ssid, string securityKey)</code>
cp	<code>int softAPNetwork(string ssid, string securityKey)</code>
cmd	<code>YWireless target softAPNetwork ssid securityKey</code>

Cette fonction ne fonctionne que sur le YoctoHub-Wireless-g et le YoctoHub-Wireless-n.

Avec le YoctoHub-Wireless-g, si une clef d'accès est configurée pour un réseau SoftAP, le réseau sera protégé par une sécurité WEP40 (5 caractères ou 10 chiffres hexadécimaux) ou WEP128 (13 caractères ou 26 chiffres hexadécimaux). Pour réduire les risques d'intrusion, il est recommandé d'utiliser une clé WEP128 basée sur 26 chiffres hexadécimaux provenant d'une bonne source aléatoire.

Avec le YoctoHub-Wireless-n, si une clef d'accès est configurée pour un réseau SoftAP, le réseau sera sécurisé par WPA2.

N'oubliez pas d'appeler la méthode `saveToFlash()` et de redémarrer le module pour que le paramètre soit appliqué.

Paramètres :

- ssid** nom du réseau sans fil à créer
- securityKey** clé d'accès de réseau, 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.

wireless→startWlanScan()**YWireless**

Déclenche un balayage des fréquences utilisable et construit la liste de réseau sans fils disponible.

js	function startWlanScan ()
cpp	int startWlanScan ()
m	-(int) startWlanScan
pas	LongInt startWlanScan (): LongInt
vb	function startWlanScan () As Integer
cs	int startWlanScan ()
java	int startWlanScan ()
uwp	async Task<int> startWlanScan ()
py	startWlanScan ()
php	function startWlanScan ()
ts	async startWlanScan (): Promise<number>
es	async startWlanScan ()
dnp	int startWlanScan ()
cp	int startWlanScan ()
cmd	YWireless target startWlanScan

Pendant le balayage des réseau sans fils, l'interface se déconnecte du réseau sans fils actuel. A la fin du balayage l'interface essaye de se reconnecter au réseau sans fil. Pendant le balayage le wlanState passe par l'état YWireless.WLANSTATE_DOWN puis par l'état YWireless.WLANSTATE_SCANNING. La liste des réseaux sans fils disponible peut être récupéré avec la méthode get_detectedWlans() à partir du moment ou get_wlanState() retourne YWireless.WLANSTATE_REJECTED ou YWireless.WLANSTATE_CONNECTED.

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

wireless→unmuteValueCallbacks()**YWireless**

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 ()
java	int unmuteValueCallbacks ()
uwp	async Task<int> unmuteValueCallbacks ()
py	unmuteValueCallbacks ()
php	function unmuteValueCallbacks ()
ts	async unmuteValueCallbacks (): Promise<number>
es	async unmuteValueCallbacks ()
dnp	int unmuteValueCallbacks ()
cp	int unmuteValueCallbacks ()
cmd	YWireless 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.

wireless→wait_async()**YWireless**

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

js	<code>function wait_async(callback, context)</code>
ts	<code>wait_async(callback: Function, context: object)</code>
es	<code>wait_async(callback, context)</code>

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.

11.3. La classe YNetwork

Interface pour interagir avec les interfaces réseau, disponibles par exemple dans le YoctoHub-Ethernet, le YoctoHub-GSM-3G-NA, le YoctoHub-GSM-4G et le YoctoHub-Wireless-n

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

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 <i>[modifiable]</i> Attente initiale avant la première notification par callback, en secondes.
network→CallbackMaxDelay <i>[modifiable]</i> Attente entre deux callback HTTP lorsque rien n'est à signaler, en secondes.
network→CallbackMethod <i>[modifiable]</i> Méthode HTTP à utiliser pour signaler les changements d'état par callback.
network→CallbackMinDelay <i>[modifiable]</i> Attente minimale entre deux callbacks HTTP, en secondes.
network→CallbackSchedule <i>[modifiable]</i> Planification des callbacks HTTP, sous forme de chaîne de caractères.
network→CallbackUrl <i>[modifiable]</i> Adresse (URL) de callback à notifier lors de changement d'état significatifs.
network→DefaultPage <i>[modifiable]</i> Page HTML à envoyer pour l'URL "/" Modifiable .
network→Discoverable <i>[modifiable]</i> état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocols uPnP/Bonjour).
network→FriendlyName <i>[lecture seule]</i> Identifiant global de la fonction au format NOM_MODULE . NOM_FONCTION.
network→FunctionId <i>[lecture seule]</i> Identifiant matériel de l'interface réseau, sans référence au module.
network→HardwareId <i>[lecture seule]</i> Identifiant matériel unique de la fonction au format SERIAL . FUNCTIONID.
network→HttpPort <i>[modifiable]</i> Port TCP utilisé pour l'interface Web du hub.
network→IpAddress <i>[lecture seule]</i> Adresse IP utilisée par le module Yoctopuce.
network→IsOnline <i>[lecture seule]</i> Vérifie si le module hébergeant la fonction est joignable, sans déclencher d'erreur.
network→LogicalName <i>[modifiable]</i> Nom logique de la fonction.
network→MacAddress <i>[lecture seule]</i> Adresse MAC de l'interface réseau, unique pour chaque module.
network→NtpServer <i>[modifiable]</i> Adresse IP du serveur de NTP à utiliser pour maintenir le module à l'heure.
network→PrimaryDNS <i>[modifiable]</i> Adresse IP du serveur de noms primaire que le module doit utiliser.
network→Readiness <i>[lecture seule]</i> état de fonctionnement atteint par l'interface réseau.
network→SecondaryDNS <i>[modifiable]</i> Adresse IP du serveur de noms secondaire que le module doit utiliser.
network→SerialNumber <i>[lecture seule]</i> Numéro de série du module, préprogrammé en usine.
network→UserPassword <i>[modifiable]</i>

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 connexion WWW tolérée (en secondes) avant de déclencher 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 (protocoles 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 connexion WWW tolérée (en secondes) avant de déclencher 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 (protocoles 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 connexion WWW tolérée (en secondes) avant de déclencher 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.FindNetwork()

YNetwork

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

js	function yFindNetwork (func)
cpp	YNetwork* FindNetwork (string func)
m	+(YNetwork*) FindNetwork : (NSString*) func
pas	TYNetwork yFindNetwork (func : string): TYNetwork
vb	function FindNetwork (ByVal func As String) As YNetwork
cs	static YNetwork FindNetwork (string func)
java	static YNetwork FindNetwork (String func)
uwp	static YNetwork FindNetwork (string func)
py	FindNetwork (func)
php	function FindNetwork (\$func)
ts	static FindNetwork (func : string): YNetwork
es	static FindNetwork (func)
dnp	static YNetworkProxy FindNetwork (string func)
cp	static YNetworkProxy * FindNetwork (string 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 où 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 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.FindNetworkInContext()

YNetwork

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

java	static YNetwork FindNetworkInContext (YAPIContext yctx , String func)
uwp	static YNetwork FindNetworkInContext (YAPIContext yctx , string func)
ts	static FindNetworkInContext (yctx : YAPIContext, func : string): YNetwork
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 où 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.FirstNetwork()

YNetwork

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

js	function yFirstNetwork ()
cpp	YNetwork * FirstNetwork ()
m	+(YNetwork*) FirstNetwork
pas	TYNetwork yFirstNetwork (): TYNetwork
vb	function FirstNetwork () As YNetwork
cs	static YNetwork FirstNetwork ()
java	static YNetwork FirstNetwork ()
uwp	static YNetwork FirstNetwork ()
py	FirstNetwork ()
php	function FirstNetwork ()
ts	static FirstNetwork (): YNetwork null
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.FirstNetworkInContext()

YNetwork

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

java	static YNetwork FirstNetworkInContext (YAPIContext yctx)
uwp	static YNetwork FirstNetworkInContext (YAPIContext yctx)
ts	static FirstNetworkInContext (yctx : YAPIContext): YNetwork null
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).

dnf	<code>static new string[] GetSimilarFunctions()</code>
-----	---

cp	<code>static vector<string> GetSimilarFunctions()</code>
----	---

Chaque chaîne retournée peut être passée en argument à la méthode `YNetwork.FindNetwork` pour obtenir un objet permettant d'interagir 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.

`dnf` `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.

`dnf` `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.

dnf [int CallbackEncoding](#)

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](#)

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.

`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 (protocoles uPnP/Bonjour).

`dnsp` `int` **Discoverable**

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'identifiant 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 exemple 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 exemple RELAYL01 - 123456 . r e l a y 1).

network→HttpPort**YNetwork**

Port TCP utilisé pour l'interface Web du hub.

dnf `int HttpPort`

Modifiable. La valeur par défaut est le port 80, utilisé habituellement par 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.

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.

`dnsp` `string NtpServer`

Modifiable. Modifie l'adresse IP du serveur NTP que le module doit utiliser. Utilisez une chaîne 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.

état de fonctionnement atteint par l'interface réseau.

dnf [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 câble 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 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 câble 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 un serveur NTP.

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.

`dnsp` `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.

dnf 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 déclencher un redémarrage automatique pour tenter de récupérer la connectivité Internet.

dnsp **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)
java	int callbackLogin (String username , String password)
py	callbackLogin (username , password)
php	function callbackLogin (\$username, \$password)
ts	async callbackLogin (username : string, password : string): Promise<number>
es	async callbackLogin (username , password)
dnp	int callbackLogin (string username , string password)
cp	int callbackLogin (string username , string 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 ()
ts	async clearCache (): Promise<void>
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 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 ()
ts	async describe (): Promise<string>
es	async describe ()

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès à 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)=RELAYL01-123456.relay1 si le module est déjà connecté ou Relay(BadCustomName.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 debugueur.

Retourne :

une chaîne de caractères décrivant l'interface réseau (ex:
Relay(MyCustomName.relay1)=RELAYL01-123456.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 ()
java	String get_adminPassword ()
uwp	async Task<string> get_adminPassword ()
py	get_adminPassword ()
php	function get_adminPassword ()
ts	async get_adminPassword (): Promise<string>
es	async get_adminPassword ()
dnf	string get_adminPassword ()
cp	string 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 `YNetwork.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 ()
java	String get_advertisedValue ()
uwp	async Task<string> get_advertisedValue ()
py	get_advertisedValue ()
php	function get_advertisedValue ()
ts	async get_advertisedValue (): Promise<string>
es	async get_advertisedValue ()
dnp	string get_advertisedValue ()
cp	string 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 `YNetwork.AVERTISEDVALUE_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 ()
java	String get_callbackCredentials ()
uwp	async Task<string> get_callbackCredentials ()
py	get_callbackCredentials ()
php	function get_callbackCredentials ()
ts	async get_callbackCredentials (): Promise<string>
es	async get_callbackCredentials ()
dnp	string get_callbackCredentials ()
cp	string 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 `YNetwork.CALLBACKCREDENTIALS_INVALID`.

network→get_callbackEncoding()**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 ()
java	int get_callbackEncoding ()
uwp	async Task<int> get_callbackEncoding ()
py	get_callbackEncoding ()
php	function get_callbackEncoding ()
ts	async get_callbackEncoding (): Promise<YNetwork_CallbackEncoding>
es	async get_callbackEncoding ()
dnp	int get_callbackEncoding ()
cp	int get_callbackEncoding ()
cmd	YNetwork target get_callbackEncoding

Retourne :

une valeur parmi YNetwork.CALLBACKENCODING_FORM, YNetwork.CALLBACKENCODING_JSON, YNetwork.CALLBACKENCODING_JSON_ARRAY, YNetwork.CALLBACKENCODING_CSV, YNetwork.CALLBACKENCODING_YOCTO_API, YNetwork.CALLBACKENCODING_JSON_NUM, YNetwork.CALLBACKENCODINGEMONCMS, YNetwork.CALLBACKENCODING_AZURE, YNetwork.CALLBACKENCODING_INFLUXDB, YNetwork.CALLBACKENCODING_MQTT, YNetwork.CALLBACKENCODING_YOCTO_API_JZON, YNetwork.CALLBACKENCODING_PRTG et YNetwork.CALLBACKENCODING_INFLUXDB_V2 représentant l'encodage à utiliser pour représenter les valeurs notifiées par callback

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

network→get_callbackInitialDelay()**YNetwork****network→callbackInitialDelay()**

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 ()
java	int get_callbackInitialDelay ()
uwp	async Task<int> get_callbackInitialDelay ()
py	get_callbackInitialDelay ()
php	function get_callbackInitialDelay ()
ts	async get_callbackInitialDelay (): Promise<number>
es	async get_callbackInitialDelay ()
dnp	int get_callbackInitialDelay ()
cp	int 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 `YNetwork.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 ()
java	int get_callbackMaxDelay ()
uwp	async Task<int> get_callbackMaxDelay ()
py	get_callbackMaxDelay ()
php	function get_callbackMaxDelay ()
ts	async get_callbackMaxDelay (): Promise<number>
es	async get_callbackMaxDelay ()
dnp	int get_callbackMaxDelay ()
cp	int 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 `YNetwork.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 ()
java	int get_callbackMethod ()
uwp	async Task<int> get_callbackMethod ()
py	get_callbackMethod ()
php	function get_callbackMethod ()
ts	async get_callbackMethod (): Promise<YNetwork_CallbackMethod>
es	async get_callbackMethod ()
dnp	int get_callbackMethod ()
cp	int get_callbackMethod ()
cmd	YNetwork target get_callbackMethod

Retourne :

une valeur parmi YNetwork.CALLBACKMETHOD_POST, YNetwork.CALLBACKMETHOD_GET et YNetwork.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 YNetwork.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 ()
java	int get_callbackMinDelay ()
uwp	async Task<int> get_callbackMinDelay ()
py	get_callbackMinDelay ()
php	function get_callbackMinDelay ()
ts	async get_callbackMinDelay (): Promise<number>
es	async get_callbackMinDelay ()
dnp	int get_callbackMinDelay ()
cp	int 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 `YNetwork.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 ()
java	String get_callbackSchedule ()
uwp	async Task<string> get_callbackSchedule ()
py	get_callbackSchedule ()
php	function get_callbackSchedule ()
ts	async get_callbackSchedule (): Promise<string>
es	async get_callbackSchedule ()
dnp	string get_callbackSchedule ()
cp	string 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 `YNetwork.CALLBACKSCHEDULE_INVALID`.

network→get_callbackUrl()**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 ()
java	String get_callbackUrl ()
uwp	async Task<string> get_callbackUrl ()
py	get_callbackUrl ()
php	function get_callbackUrl ()
ts	async get_callbackUrl (): Promise<string>
es	async get_callbackUrl ()
dnp	string get_callbackUrl ()
cp	string 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 YNetwork.CALLBACKURL_INVALID.

network→get_defaultPage()**YNetwork****network→defaultPage()**

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 ()
java	String get_defaultPage ()
uwp	async Task<string> get_defaultPage ()
py	get_defaultPage ()
php	function get_defaultPage ()
ts	async get_defaultPage (): Promise<string>
es	async get_defaultPage ()
dnp	string get_defaultPage ()
cp	string 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 YNetwork.DEFAULTPAGE_INVALID.

network→get_discoverable()**YNetwork****network→discoverable()**

Retourne l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocoles 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 ()
java	int get_discoverable ()
uwp	async Task<int> get_discoverable ()
py	get_discoverable ()
php	function get_discoverable ()
ts	async get_discoverable (): Promise<YNetwork_Discoverable>
es	async get_discoverable ()
dnp	int get_discoverable ()
cp	int get_discoverable ()
cmd	YNetwork target get_discoverable

Retourne :

soit `YNetwork.DISCOVERABLE_FALSE`, soit `YNetwork.DISCOVERABLE_TRUE`, selon l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocoles uPnP/Bonjour)

En cas d'erreur, déclenche une exception ou retourne `YNetwork.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 ()
ts	get_errorMessage (): string
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()**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 ()
ts	get_errorType (): number
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()**YNetwork****network→friendlyName()**

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 ()
java	String get_friendlyName ()
py	get_friendlyName ()
php	function get_friendlyName ()
ts	async get_friendlyName (): Promise<string>
es	async get_friendlyName ()
dnp	string get_friendlyName ()
cp	string 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'identifiant 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 `YNetwork.FRIENDLYNAME_INVALID`.

network→**get_functionDescriptor()****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 ()
ts	async get_functionDescriptor (): Promise<string>
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\$CLASSNAME\$.FUNCTIONDESCRIPTOR_INVALID

network→get_functionId()**YNetwork****network→functionId()**

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 ()
java	String get_functionId ()
py	get_functionId ()
php	function get_functionId ()
ts	async get_functionId (): Promise<string>
es	async get_functionId ()
dnp	string get_functionId ()
cp	string get_functionId ()

Par exemple `relay1`.

Retourne :

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

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

network→**get_hardwareId()****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 ()
java	String get_hardwareId ()
py	get_hardwareId ()
php	function get_hardwareId ()
ts	async get_hardwareId (): Promise<string>
es	async get_hardwareId ()
dnp	string get_hardwareId ()
cp	string get_hardwareId ()

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel de l'interface réseau (par exemple RELAYL01-123456 . relay1).

Retourne :

une chaîne de caractères identifiant l'interface réseau (ex: RELAYL01-123456 . relay1)

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

network→get_httpPort()**YNetwork****network→httpPort()**

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 ()
java	int get_httpPort ()
uwp	async Task<int> get_httpPort ()
py	get_httpPort ()
php	function get_httpPort ()
ts	async get_httpPort (): Promise<number>
es	async get_httpPort ()
dnp	int get_httpPort ()
cp	int 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 YNetwork.HTTPPORT_INVALID.

network→**get_ipAddress()****YNetwork****network**→**ipAddress()**

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 ()
java	String get_ipAddress ()
uwp	async Task<string> get_ipAddress ()
py	get_ipAddress ()
php	function get_ipAddress ()
ts	async get_ipAddress (): Promise<string>
es	async get_ipAddress ()
dnp	string get_ipAddress ()
cp	string 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 `YNetwork.IPADDRESS_INVALID`.

network→get_ipConfig()**YNetwork****network→ipConfig()**

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 ()
java	String get_ipConfig ()
uwp	async Task<string> get_ipConfig ()
py	get_ipConfig ()
php	function get_ipConfig ()
ts	async get_ipConfig (): Promise<string>
es	async get_ipConfig ()
dnp	string get_ipConfig ()
cp	string 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 par 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 `YNetwork . 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 ()
java	String get_logicalName ()
uwp	async Task<string> get_logicalName ()
py	get_logicalName ()
php	function get_logicalName ()
ts	async get_logicalName (): Promise<string>
es	async get_logicalName ()
dnp	string get_logicalName ()
cp	string 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 `YNetwork.LOGICALNAME_INVALID`.

network→get_macAddress()**YNetwork****network→macAddress()**

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 ()
java	String get_macAddress ()
uwp	async Task<string> get_macAddress ()
py	get_macAddress ()
php	function get_macAddress ()
ts	async get_macAddress (): Promise<string>
es	async get_macAddress ()
dnp	string get_macAddress ()
cp	string 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 `YNetwork.MACADDRESS_INVALID`.

network→get_module()**network→module()**

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 ()
java	YModule get_module ()
py	get_module ()
php	function get_module ()
ts	async get_module (): Promise<YModule>
es	async get_module ()
dnp	YModuleProxy get_module ()
cp	YModuleProxy * 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()**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 ()
java	String get_ntpServer ()
uwp	async Task<string> get_ntpServer ()
py	get_ntpServer ()
php	function get_ntpServer ()
ts	async get_ntpServer (): Promise<string>
es	async get_ntpServer ()
dnp	string get_ntpServer ()
cp	string 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 `YNetwork.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 ()
java	int get_poeCurrent ()
uwp	async Task<int> get_poeCurrent ()
py	get_poeCurrent ()
php	function get_poeCurrent ()
ts	async get_poeCurrent (): Promise<number>
es	async get_poeCurrent ()
dnp	int get_poeCurrent ()
cp	int 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 `YNetwork.POECURRENT_INVALID`.

network→get_primaryDNS()**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 ()
java	String get_primaryDNS ()
uwp	async Task<string> get_primaryDNS ()
py	get_primaryDNS ()
php	function get_primaryDNS ()
ts	async get_primaryDNS (): Promise<string>
es	async get_primaryDNS ()
dnp	string get_primaryDNS ()
cp	string 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 YNetwork.PRIMARYDNS_INVALID.

network→get_readiness()**YNetwork****network→readiness()**

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 ()
java	int get_readiness ()
uwp	async Task<int> get_readiness ()
py	get_readiness ()
php	function get_readiness ()
ts	async get_readiness (): Promise<YNetwork_Readiness>
es	async get_readiness ()
dnp	int get_readiness ()
cp	int 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 câble 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 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 câble est connecté aux deux bouts. Pour une connection à un point d'accès réseau sans fil, il démontre que les paramètres de sécurité 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 un serveur NTP.

Retourne :

une valeur parmi YNetwork.READINESS_DOWN, YNetwork.READINESS_EXISTS, YNetwork.READINESS_LINKED, YNetwork.READINESS_LAN_OK et YNetwork.READINESS_WWW_OK représentant l'état de fonctionnement atteint par l'interface réseau

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

network→get_router()**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 ()
java	String get_router ()
uwp	async Task<string> get_router ()
py	get_router ()
php	function get_router ()
ts	async get_router (): Promise<string>
es	async get_router ()
dnp	string get_router ()
cp	string 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 YNetwork.ROUTER_INVALID.

network→get_secondaryDNS()**YNetwork****network→secondaryDNS()**

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 ()
java	String get_secondaryDNS ()
uwp	async Task<string> get_secondaryDNS ()
py	get_secondaryDNS ()
php	function get_secondaryDNS ()
ts	async get_secondaryDNS (): Promise<string>
es	async get_secondaryDNS ()
dnp	string get_secondaryDNS ()
cp	string 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 `YNetwork.SECONDARYDNS_INVALID`.

network→**get_serialNumber()****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 ()
java	String get_serialNumber ()
uwp	async Task<string> get_serialNumber ()
py	get_serialNumber ()
php	function get_serialNumber ()
ts	async get_serialNumber (): Promise<string>
es	async get_serialNumber ()
dnp	string get_serialNumber ()
cp	string 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 YFunction.SERIALNUMBER_INVALID.

network→get_subnetMask()**YNetwork****network→subnetMask()**

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 ()
java	String get_subnetMask ()
uwp	async Task<string> get_subnetMask ()
py	get_subnetMask ()
php	function get_subnetMask ()
ts	async get_subnetMask (): Promise<string>
es	async get_subnetMask ()
dnp	string get_subnetMask ()
cp	string 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 YNetwork.SUBNETMASK_INVALID.

network→get_userdata()**network→userdata()**

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

js	<code>function get_userdata()</code>
cpp	<code>void * get_userdata()</code>
m	<code>-(id) userData</code>
pas	<code>Tobject get_userdata(): Tobject</code>
vb	<code>function get_userdata() As Object</code>
cs	<code>object get_userdata()</code>
java	<code>Object get_userdata()</code>
py	<code>get_userdata()</code>
php	<code>function get_userdata()</code>
ts	<code>async get_userdata(): Promise<object null></code>
es	<code>async get_userdata()</code>

Cet attribut n'est 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 ()
java	String get_userPassword ()
uwp	async Task<string> get_userPassword ()
py	get_userPassword ()
php	function get_userPassword ()
ts	async get_userPassword (): Promise<string>
es	async get_userPassword ()
dnf	string get_userPassword ()
cp	string 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 `YNetwork.USERPASSWORD_INVALID`.

network→get_wwwWatchdogDelay()**YNetwork****network→wwwWatchdogDelay()**

Retourne la durée de perte de connection WWW tolérée (en secondes) avant de déclencher 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 ()
java	int get_wwwWatchdogDelay ()
uwp	async Task<int> get_wwwWatchdogDelay ()
py	get_wwwWatchdogDelay ()
php	function get_wwwWatchdogDelay ()
ts	async get_wwwWatchdogDelay (): Promise<number>
es	async get_wwwWatchdogDelay ()
dnp	int get_wwwWatchdogDelay ()
cp	int 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 déclencher un redémarrage automatique pour tenter de récupérer la connectivité Internet

En cas d'erreur, déclenche une exception ou retourne `YNetwork.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 ()
java	boolean isOnline ()
py	isOnline ()
php	function isOnline ()
ts	async isOnline (): Promise<boolean>
es	async isOnline ()
dnp	bool isOnline ()
cp	bool 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	<code>bool isReadOnly()</code>
m	<code>-(bool) isReadOnly</code>
pas	<code>boolean isReadOnly()</code> : boolean
vb	<code>function isReadOnly()</code> As Boolean
cs	<code>bool isReadOnly()</code>
java	<code>boolean isReadOnly()</code>
uwp	<code>async Task<bool> isReadOnly()</code>
py	<code>isReadOnly()</code>
php	<code>function isReadOnly()</code>
ts	<code>async isReadOnly()</code> : Promise<boolean>
es	<code>async isReadOnly()</code>
dnp	<code>bool isReadOnly()</code>
cp	<code>bool isReadOnly()</code>
cmd	<code>YNetwork target isReadOnly</code>

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)
ts	async load (msValidity : number): Promise<number>
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 occasionnellement 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(ByVal attrName As String) As String
cs	string loadAttribute(string attrName)
java	String loadAttribute(String attrName)
uwp	async Task<string> loadAttribute(string attrName)
py	loadAttribute(attrName)
php	function loadAttribute(\$attrName)
ts	async loadAttribute(attrName: string): Promise<string>
es	async loadAttribute(attrName)
dnf	string loadAttribute(string attrName)
cp	string loadAttribute(string 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 occasionnellement 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 ()
java	int muteValueCallbacks ()
uwp	async Task<int> muteValueCallbacks ()
py	muteValueCallbacks ()
php	function muteValueCallbacks ()
ts	async muteValueCallbacks (): Promise<number>
es	async muteValueCallbacks ()
dnp	int muteValueCallbacks ()
cp	int 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 déclenchement 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	-(nullable 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 ()
ts	nextNetwork (): YNetwork null
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 (ByVal host As String) As String
cs	string ping (string host)
java	String ping (String host)
uwp	async Task<string> ping (string host)
py	ping (host)
php	function ping (\$ host)
ts	async ping (host : string): Promise<string>
es	async ping (host)
dnp	string ping (string host)
cp	string ping (string 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 _Nullable) callback
pas	LongInt registerValueCallback (callback : TYNetworkValueCallback): LongInt
vb	function registerValueCallback (ByVal callback As YNetworkValueCallback) 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)
ts	async registerValueCallback (callback : YNetworkValueCallback null): Promise<number>
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)
java	int set_adminPassword (String newval)
uwp	async Task<int> set_adminPassword (string newval)
py	set_adminPassword (newval)
php	function set_adminPassword (\$ newval)
ts	async set_adminPassword (newval : string): Promise<number>
es	async set_adminPassword (newval)
dnp	int set_adminPassword (string newval)
cp	int set_adminPassword (string 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()**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)
java	int set_callbackCredentials (String newval)
uwp	async Task<int> set_callbackCredentials (string newval)
py	set_callbackCredentials (newval)
php	function set_callbackCredentials (\$newval)
ts	async set_callbackCredentials (newval : string): Promise<number>
es	async set_callbackCredentials (newval)
dnp	int set_callbackCredentials (string newval)
cp	int set_callbackCredentials (string 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)
java	int set_callbackEncoding (int newval)
uwp	async Task<int> set_callbackEncoding (int newval)
py	set_callbackEncoding (newval)
php	function set_callbackEncoding (\$ newval)
ts	async set_callbackEncoding (newval : YNetwork_CallbackEncoding): Promise<number>
es	async set_callbackEncoding (newval)
dnp	int set_callbackEncoding (int newval)
cp	int set_callbackEncoding (int 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 `YNetwork.CALLBACKENCODING_FORM`,
`YNetwork.CALLBACKENCODING_JSON`,
`YNetwork.CALLBACKENCODING_JSON_ARRAY`,
`YNetwork.CALLBACKENCODING_CSV`,
`YNetwork.CALLBACKENCODING_YOCTO_API`,
`YNetwork.CALLBACKENCODING_JSON_NUM`,
`YNetwork.CALLBACKENCODING_EMONCMS`,
`YNetwork.CALLBACKENCODING_AZURE`,
`YNetwork.CALLBACKENCODING_INFLUXDB`,
`YNetwork.CALLBACKENCODING_MQTT`,
`YNetwork.CALLBACKENCODING_YOCTO_API_JZON`,
`YNetwork.CALLBACKENCODING_PRTG` et
`YNetwork.CALLBACKENCODING_INFLUXDB_V2` 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)
java	int set_callbackInitialDelay (int newval)
uwp	async Task<int> set_callbackInitialDelay (int newval)
py	set_callbackInitialDelay (newval)
php	function set_callbackInitialDelay (\$newval)
ts	async set_callbackInitialDelay (newval : number): Promise<number>
es	async set_callbackInitialDelay (newval)
dnp	int set_callbackInitialDelay (int newval)
cp	int set_callbackInitialDelay (int 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)
java	int set_callbackMaxDelay (int newval)
uwp	async Task<int> set_callbackMaxDelay (int newval)
py	set_callbackMaxDelay (newval)
php	function set_callbackMaxDelay (\$newval)
ts	async set_callbackMaxDelay (newval : number): Promise<number>
es	async set_callbackMaxDelay (newval)
dnp	int set_callbackMaxDelay (int newval)
cp	int set_callbackMaxDelay (int 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)
java	int set_callbackMethod (int newval)
uwp	async Task<int> set_callbackMethod (int newval)
py	set_callbackMethod (newval)
php	function set_callbackMethod (\$ newval)
ts	async set_callbackMethod (newval : YNetwork_CallbackMethod): Promise<number>
es	async set_callbackMethod (newval)
dnp	int set_callbackMethod (int newval)
cp	int set_callbackMethod (int 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 `YNetwork.CALLBACKMETHOD_POST`, `YNetwork.CALLBACKMETHOD_GET` et `YNetwork.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)
java	int set_callbackMinDelay (int newval)
uwp	async Task<int> set_callbackMinDelay (int newval)
py	set_callbackMinDelay (newval)
php	function set_callbackMinDelay (\$ newval)
ts	async set_callbackMinDelay (newval : number): Promise<number>
es	async set_callbackMinDelay (newval)
dnp	int set_callbackMinDelay (int newval)
cp	int set_callbackMinDelay (int 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)
java	int set_callbackSchedule (String newval)
uwp	async Task<int> set_callbackSchedule (string newval)
py	set_callbackSchedule (newval)
php	function set_callbackSchedule (\$ newval)
ts	async set_callbackSchedule (newval : string): Promise<number>
es	async set_callbackSchedule (newval)
dnp	int set_callbackSchedule (string newval)
cp	int set_callbackSchedule (string 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)
java	int set_callbackUrl (String newval)
uwp	async Task<int> set_callbackUrl (string newval)
py	set_callbackUrl (newval)
php	function set_callbackUrl (\$ newval)
ts	async set_callbackUrl (newval : string): Promise<number>
es	async set_callbackUrl (newval)
dnp	int set_callbackUrl (string newval)
cp	int set_callbackUrl (string 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()**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)
java	int set_defaultPage (String newval)
uwp	async Task<int> set_defaultPage (string newval)
py	set_defaultPage (newval)
php	function set_defaultPage (\$ newval)
ts	async set_defaultPage (newval : string): Promise<number>
es	async set_defaultPage (newval)
dnp	int set_defaultPage (string newval)
cp	int set_defaultPage (string 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 (protocoles 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)
java	int set_discoverable (int newval)
uwp	async Task<int> set_discoverable (int newval)
py	set_discoverable (newval)
php	function set_discoverable (\$ newval)
ts	async set_discoverable (newval : YNetwork_Discoverable): Promise<number>
es	async set_discoverable (newval)
dnp	int set_discoverable (int newval)
cp	int set_discoverable (int 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 `YNetwork.DISCOVERABLE_FALSE`, soit `YNetwork.DISCOVERABLE_TRUE`, selon l'état d'activation du protocole d'annonce sur le réseau permettant de retrouver facilement le module (protocoles 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()**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)
java	int set_httpPort (int newval)
uwp	async Task<int> set_httpPort (int newval)
py	set_httpPort (newval)
php	function set_httpPort (\$newval)
ts	async set_httpPort (newval : number): Promise<number>
es	async set_httpPort (newval)
dnp	int set_httpPort (int newval)
cp	int set_httpPort (int newval)
cmd	YNetwork target set_httpPort newval

La valeur par défaut est le port 80, utilisé habituellement par 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)
java	int set_logicalName (String newval)
uwp	async Task<int> set_logicalName (string newval)
py	set_logicalName (newval)
php	function set_logicalName (\$ newval)
ts	async set_logicalName (newval : string): Promise<number>
es	async set_logicalName (newval)
dnp	int set_logicalName (string newval)
cp	int set_logicalName (string 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()**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)
java	int set_ntpServer (String newval)
uwp	async Task<int> set_ntpServer (string newval)
py	set_ntpServer (newval)
php	function set_ntpServer (\$newval)
ts	async set_ntpServer (newval : string): Promise<number>
es	async set_ntpServer (newval)
dnp	int set_ntpServer (string newval)
cp	int set_ntpServer (string newval)
cmd	YNetwork target set_ntpServer newval

Utilisez une chaîne 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	<code>function set_periodicCallbackSchedule(interval, offset)</code>
cpp	<code>int set_periodicCallbackSchedule(string interval, int offset)</code>
m	<code>-(int) setPeriodicCallbackSchedule : (NSString*) interval : (int) offset</code>
pas	<code>LongInt set_periodicCallbackSchedule(interval: string, offset: LongInt): LongInt</code>
vb	<code>function set_periodicCallbackSchedule(ByVal interval As String, ByVal offset As Integer) As Integer</code>
cs	<code>int set_periodicCallbackSchedule(string interval, int offset)</code>
java	<code>int set_periodicCallbackSchedule(String interval, int offset)</code>
uwp	<code>async Task<int> set_periodicCallbackSchedule(string interval, int offset)</code>
py	<code>set_periodicCallbackSchedule(interval, offset)</code>
php	<code>function set_periodicCallbackSchedule(\$interval, \$offset)</code>
ts	<code>async set_periodicCallbackSchedule(interval: string, offset: number): Promise<number></code>
es	<code>async set_periodicCallbackSchedule(interval, offset)</code>
dnp	<code>int set_periodicCallbackSchedule(string interval, int offset)</code>
cp	<code>int set_periodicCallbackSchedule(string interval, int offset)</code>
cmd	<code>YNetwork target set_periodicCallbackSchedule interval offset</code>

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()**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)
java	int set_primaryDNS (String newval)
uwp	async Task<int> set_primaryDNS (string newval)
py	set_primaryDNS (newval)
php	function set_primaryDNS (\$ newval)
ts	async set_primaryDNS (newval : string): Promise<number>
es	async set_primaryDNS (newval)
dnp	int set_primaryDNS (string newval)
cp	int set_primaryDNS (string 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)
java	int set_secondaryDNS (String newval)
uwp	async Task<int> set_secondaryDNS (string newval)
py	set_secondaryDNS (newval)
php	function set_secondaryDNS (\$ newval)
ts	async set_secondaryDNS (newval : string): Promise<number>
es	async set_secondaryDNS (newval)
dnp	int set_secondaryDNS (string newval)
cp	int set_secondaryDNS (string 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()**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)
ts	async set_userdata (data : object null): Promise<void>
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)
java	int set_userPassword (String newval)
uwp	async Task<int> set_userPassword (string newval)
py	set_userPassword (newval)
php	function set_userPassword (\$ newval)
ts	async set_userPassword (newval : string): Promise<number>
es	async set_userPassword (newval)
dnp	int set_userPassword (string newval)
cp	int set_userPassword (string 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 déclencher 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)
java	int set_wwwWatchdogDelay (int newval)
uwp	async Task<int> set_wwwWatchdogDelay (int newval)
py	set_wwwWatchdogDelay (newval)
php	function set_wwwWatchdogDelay (\$newval)
ts	async set_wwwWatchdogDelay (newval : number): Promise<number>
es	async set_wwwWatchdogDelay (newval)
dnp	int set_wwwWatchdogDelay (int newval)
cp	int set_wwwWatchdogDelay (int 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 déclencher 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 ()
java	int triggerCallback ()
uwp	async Task<int> triggerCallback ()
py	triggerCallback ()
php	function triggerCallback ()
ts	async triggerCallback (): Promise<number>
es	async triggerCallback ()
dnp	int triggerCallback ()
cp	int triggerCallback ()
cmd	YNetwork target triggerCallback

Cette fonction peut même être appelée 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 ()
java	int unmuteValueCallbacks ()
uwp	async Task<int> unmuteValueCallbacks ()
py	unmuteValueCallbacks ()
php	function unmuteValueCallbacks ()
ts	async unmuteValueCallbacks (): Promise<number>
es	async unmuteValueCallbacks ()
dnp	int unmuteValueCallbacks ()
cp	int 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	<code>function useDHCP(fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)</code>
cpp	<code>int useDHCP(string fallbackIpAddr, int fallbackSubnetMaskLen, string fallbackRouter)</code>
m	<code>-(int) useDHCP : (NSString*) fallbackIpAddr : (int) fallbackSubnetMaskLen : (NSString*) fallbackRouter</code>
pas	<code>LongInt useDHCP(fallbackIpAddr: string, fallbackSubnetMaskLen: LongInt, fallbackRouter: string): LongInt</code>
vb	<code>function useDHCP(ByVal fallbackIpAddr As String, ByVal fallbackSubnetMaskLen As Integer, ByVal fallbackRouter As String) As Integer</code>
cs	<code>int useDHCP(string fallbackIpAddr, int fallbackSubnetMaskLen, string fallbackRouter)</code>
java	<code>int useDHCP(String fallbackIpAddr, int fallbackSubnetMaskLen, String fallbackRouter)</code>
uwp	<code>async Task<int> useDHCP(string fallbackIpAddr, int fallbackSubnetMaskLen, string fallbackRouter)</code>
py	<code>useDHCP(fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)</code>
php	<code>function useDHCP(\$fallbackIpAddr, \$fallbackSubnetMaskLen, \$fallbackRouter)</code>
ts	<code>async useDHCP(fallbackIpAddr: string, fallbackSubnetMaskLen: number, fallbackRouter: string): Promise<number></code>
es	<code>async useDHCP(fallbackIpAddr, fallbackSubnetMaskLen, fallbackRouter)</code>
dnp	<code>int useDHCP(string fallbackIpAddr, int fallbackSubnetMaskLen, string fallbackRouter)</code>
cp	<code>int useDHCP(string fallbackIpAddr, int fallbackSubnetMaskLen, string fallbackRouter)</code>
cmd	<code>YNetwork target useDHCP fallbackIpAddr fallbackSubnetMaskLen fallbackRouter</code>

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 ()
java	int useDHCPauto ()
uwp	async Task<int> useDHCPauto ()
py	useDHCPauto ()
php	function useDHCPauto ()
ts	async useDHCPauto (): Promise<number>
es	async useDHCPauto ()
dnp	int useDHCPauto ()
cp	int 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( ByVal ipAddress As String,
    ByVal subnetMaskLen As Integer,
    ByVal router As String) As Integer
cs 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)
ts async useStaticIP( ipAddress: string, subnetMaskLen: number, router: string): Promise<number>
es async useStaticIP( ipAddress, subnetMaskLen, router)
dnp int useStaticIP( string ipAddress,
    int subnetMaskLen,
    string router)
cp int useStaticIP( string ipAddress,
    int subnetMaskLen,
    string 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	<code>function wait_async(callback, context)</code>
ts	<code>wait_async(callback: Function, context: object)</code>
es	<code>wait_async(callback, context)</code>

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.

11.4. La classe YFiles

Interface pour interagir avec les systèmes de fichier, disponibles par exemple dans le Yocto-Color-V2, le Yocto-Serial, le YoctoHub-Ethernet et le YoctoHub-Wireless-n

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
java	import com.yoctopuce.YoctoAPI.YFiles;
uwp	import com.yoctopuce.YoctoAPI.YFiles;
py	from yocto_files import *
php	require_once('yocto_files.php');
ts	in HTML: import { YFiles } from '../dist/esm/yocto_files.js'; in Node.js: import { YFiles } from 'yoctolib-cjs/yocto_files.js';
es	in HTML: <script src="../../lib/yocto_files.js"></script> in node.js: require('yoctolib-es2017/yocto_files.js');
dnf	import YoctoProxyAPI.YFilesProxy
cp	#include "yocto_files_proxy.h"
vi	YFiles.vi
mL	import YoctoProxyAPI.YFilesProxy

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.FindFiles()

YFiles

Permet de retrouver un système de fichier d'après un identifiant donné.

js	function yFindFiles (func)
cpp	YFiles* FindFiles (string func)
m	+(YFiles*) FindFiles : (NSString*) func
pas	TYFiles yFindFiles (func : string): TYFiles
vb	function FindFiles (ByVal func As String) As YFiles
cs	static YFiles FindFiles (string func)
java	static YFiles FindFiles (String func)
uwp	static YFiles FindFiles (string func)
py	FindFiles (func)
php	function FindFiles (\$func)
ts	static FindFiles (func : string): YFiles
es	static FindFiles (func)
dnp	static YFilesProxy FindFiles (string func)
cp	static YFilesProxy * FindFiles (string 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 où 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 `YRGBLED2.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)
ts	static FindFilesInContext (yctx : YAPIContext, func : string): YFiles
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 où 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 `YRGBLED2.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 * FirstFiles ()
m	+(YFiles*) FirstFiles
pas	TYFiles yFirstFiles (): TYFiles
vb	function FirstFiles () As YFiles
cs	static YFiles FirstFiles ()
java	static YFiles FirstFiles ()
uwp	static YFiles FirstFiles ()
py	FirstFiles ()
php	function FirstFiles ()
ts	static FirstFiles (): YFiles null
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)
ts	static FirstFilesInContext (yctx : YAPIContext): YFiles null
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	<code>static new string[] GetSimilarFunctions()</code>
-----	--

cp	<code>static vector<string> GetSimilarFunctions()</code>
----	--

Chaque chaîne retournée peut être passée en argument à la méthode `YFiles.FindFiles` pour obtenir un objet permettant d'interagir 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`.

dnf	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'identifiant 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 exemple `re`lay1`.

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 exemple RELAYL01-123456 . r e l a y 1).

files→IsOnline**YFiles**

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

dnf **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.

`dnf` `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 ()
ts	async clearCache (): Promise<void>
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 ()
ts	async describe (): Promise<string>
es	async describe ()

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès à 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 méthode va retourner `Relay(MyCustomName.relay1)=RELAYL01-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

Retourne :

une chaîne de caractères décrivant le système de fichier (ex:
`Relay(MyCustomName.relay1)=RELAYL01-123456.relay1`)

files→download()

YFiles

Télécharge le fichier choisi du filesystem 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 (ByVal pathname As String) As Byte
cs	byte[] download (string pathname)
java	byte[] download (String pathname)
uwp	async Task<byte[]> download (string pathname)
py	download (pathname)
php	function download (\$pathname)
ts	async download (pathname : string): Promise<Uint8Array>
es	async download (pathname)
dnp	byte[] download (string pathname)
cp	string download (string 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 existe 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 (ByVal filename As String) As Boolean
cs	bool fileExist (string filename)
java	boolean fileExist (String filename)
uwp	async Task<bool> fileExist (string filename)
py	fileExist (filename)
php	function fileExist (\$ filename)
ts	async fileExist (filename : string): Promise<boolean>
es	async fileExist (filename)
dnp	bool fileExist (string filename)
cp	bool fileExist (string filename)
cmd	YFiles target fileExist filename

Paramètres :

filename le nom de fichier.

Retourne :

vrai si le fichier existe, et faux si 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 ()
java	int format_fs ()
uwp	async Task<int> format_fs ()
py	format_fs ()
php	function format_fs ()
ts	async format_fs (): Promise<number>
es	async format_fs ()
dnp	int format_fs ()
cp	int 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()**

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 ()
java	String get_advertisedValue ()
uwp	async Task<string> get_advertisedValue ()
py	get_advertisedValue ()
php	function get_advertisedValue ()
ts	async get_advertisedValue (): Promise<string>
es	async get_advertisedValue ()
dnp	string get_advertisedValue ()
cp	string 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 `YFiles.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 ()
ts	get_errorMessage (): string
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()**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 ()
ts	get_errorType (): number
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()**YFiles****files→filesCount()**

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 ()
java	int get_filesCount ()
uwp	async Task<int> get_filesCount ()
py	get_filesCount ()
php	function get_filesCount ()
ts	async get_filesCount (): Promise<number>
es	async get_filesCount ()
dnp	int get_filesCount ()
cp	int 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 `YFiles.FILES_COUNT_INVALID`.

files→get_freeSpace()**files→freeSpace()**

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 ()
java	int get_freeSpace ()
uwp	async Task<int> get_freeSpace ()
py	get_freeSpace ()
php	function get_freeSpace ()
ts	async get_freeSpace (): Promise<number>
es	async get_freeSpace ()
dnp	int get_freeSpace ()
cp	int 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 `YFiles.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 ()
java	String get_friendlyName ()
py	get_friendlyName ()
php	function get_friendlyName ()
ts	async get_friendlyName (): Promise<string>
es	async get_friendlyName ()
dnp	string get_friendlyName ()
cp	string 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'identifiant 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 `YFiles.FRIENDLYNAME_INVALID`.

files→get_functionDescriptor()**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 ()
ts	async get_functionDescriptor (): Promise<string>
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\$CLASSNAME\$.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 ()
java	String get_functionId ()
py	get_functionId ()
php	function get_functionId ()
ts	async get_functionId (): Promise<string>
es	async get_functionId ()
dnp	string get_functionId ()
cp	string get_functionId ()

Par exemple `reLay1`.

Retourne :

une chaîne de caractères identifiant le système de fichier (ex: `reLay1`)

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

files→**get_hardwareId()****files**→**hardwareId()**

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 ()
java	String get_hardwareId ()
py	get_hardwareId ()
php	function get_hardwareId ()
ts	async get_hardwareId (): Promise<string>
es	async get_hardwareId ()
dnp	string get_hardwareId ()
cp	string get_hardwareId ()

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du système de fichier (par exemple RELAYL01-123456 . r e l a y 1).

Retourne :

une chaîne de caractères identifiant le système de fichier (ex: RELAYL01-123456 . r e l a y 1)

En cas d'erreur, déclenche une exception ou retourne YF i l e s . 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 (ByVal pattern As String) As List
cs	List<YFileRecord> 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)
ts	async get_list (pattern : string): Promise<YFileRecord[]
es	async get_list (pattern)
dnp	YFileRecordProxy[] get_list (string pattern)
cp	vector<YFileRecordProxy> get_list (string 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()**

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 ()
java	String get_logicalName ()
uwp	async Task<string> get_logicalName ()
py	get_logicalName ()
php	function get_logicalName ()
ts	async get_logicalName (): Promise<string>
es	async get_logicalName ()
dnp	string get_logicalName ()
cp	string 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 YFiles.LOGICALNAME_INVALID.

files→get_module()**YFiles****files→module()**

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 ()
java	YModule get_module ()
py	get_module ()
php	function get_module ()
ts	async get_module (): Promise<YModule>
es	async get_module ()
dnp	YModuleProxy get_module ()
cp	YModuleProxy * 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()**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 ()
java	String get_serialNumber ()
uwp	async Task<string> get_serialNumber ()
py	get_serialNumber ()
php	function get_serialNumber ()
ts	async get_serialNumber (): Promise<string>
es	async get_serialNumber ()
dnp	string get_serialNumber ()
cp	string 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 YFunction.SERIALNUMBER_INVALID.

files→get_userdata()**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 ()
ts	async get_userdata (): Promise<object null>
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 ()
java	boolean isOnline ()
py	isOnline ()
php	function isOnline ()
ts	async isOnline (): Promise<boolean>
es	async isOnline ()
dnp	bool isOnline ()
cp	bool 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	<code>bool isReadOnly()</code>
m	<code>-(bool) isReadOnly</code>
pas	<code>boolean isReadOnly()</code> : boolean
vb	<code>function isReadOnly()</code> As Boolean
cs	<code>bool isReadOnly()</code>
java	<code>boolean isReadOnly()</code>
uwp	<code>async Task<bool> isReadOnly()</code>
py	<code>isReadOnly()</code>
php	<code>function isReadOnly()</code>
ts	<code>async isReadOnly()</code> : Promise<boolean>
es	<code>async isReadOnly()</code>
dnp	<code>bool isReadOnly()</code>
cp	<code>bool isReadOnly()</code>
cmd	YFiles target isReadOnly

Retourne vrai si la fonction est protégé en écriture ou que la fonction n'est pas disponible.

Retourne :

`true` si la fonction est protégé en écriture ou que la fonction 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)
ts	async load (msValidity : number): Promise<number>
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 occasionnellement 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(ByVal attrName As String) As String
cs	string loadAttribute(string attrName)
java	String loadAttribute(String attrName)
uwp	async Task<string> loadAttribute(string attrName)
py	loadAttribute(attrName)
php	function loadAttribute(\$attrName)
ts	async loadAttribute(attrName: string): Promise<string>
es	async loadAttribute(attrName)
dnf	string loadAttribute(string attrName)
cp	string loadAttribute(string 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 occasionnellement 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-qu'el 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 ()
java	int muteValueCallbacks ()
uwp	async Task<int> muteValueCallbacks ()
py	muteValueCallbacks ()
php	function muteValueCallbacks ()
ts	async muteValueCallbacks (): Promise<number>
es	async muteValueCallbacks ()
dnp	int muteValueCallbacks ()
cp	int 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 déclenchement 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	-(nullable 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 ()
ts	nextFiles (): YFiles null
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 _Nullable) callback
pas	LongInt registerValueCallback (callback : TYFilesValueCallback): LongInt
vb	function registerValueCallback (ByVal callback As YFilesValueCallback) 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)
ts	async registerValueCallback (callback : YFilesValueCallback null): Promise<number>
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'objet 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 (ByVal pathname As String) As Integer
cs	int remove (string pathname)
java	int remove (String pathname)
uwp	async Task<int> remove (string pathname)
py	remove (pathname)
php	function remove (\$pathname)
ts	async remove (pathname : string): Promise<number>
es	async remove (pathname)
dnp	int remove (string pathname)
cp	int remove (string 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 occupe. 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)
java	int set_logicalName (String newval)
uwp	async Task<int> set_logicalName (string newval)
py	set_logicalName (newval)
php	function set_logicalName (\$ newval)
ts	async set_logicalName (newval : string): Promise<number>
es	async set_logicalName (newval)
dnp	int set_logicalName (string newval)
cp	int set_logicalName (string 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()**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 getUserData.

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)
ts	async set_userData (data : object null): Promise<void>
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 ()
java	int unmuteValueCallbacks ()
uwp	async Task<int> unmuteValueCallbacks ()
py	unmuteValueCallbacks ()
php	function unmuteValueCallbacks ()
ts	async unmuteValueCallbacks (): Promise<number>
es	async unmuteValueCallbacks ()
dnp	int unmuteValueCallbacks ()
cp	int 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 (ByVal pathname As String, ByVal content As Byte())
cs	int upload (string pathname , byte[] content)
java	int upload (String pathname , byte[] content)
uwp	async Task<int> upload (string pathname , byte[] content)
py	upload (pathname , content)
php	function upload (\$pathname, \$content)
ts	async upload (pathname : string, content : Uint8Array): Promise<number>
es	async upload (pathname , content)
dnp	int upload (string pathname , byte[] content)
cp	int upload (string pathname , string 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	<code>function wait_async(callback, context)</code>
ts	<code>wait_async(callback: Function, context: object)</code>
es	<code>wait_async(callback, context)</code>

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-qu'il est à la fonction de callback

Retourne :

rien du tout.

11.5. La classe YRealTimeClock

Interface pour interagir avec les horloges à temps réel, disponibles par exemple dans le YoctoHub-GSM-3G-EU, le YoctoHub-GSM-3G-NA, le YoctoHub-GSM-4G et le YoctoHub-Wireless-n

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: <code><script src="../../lib/yocto_realtimeclock.js"></script></code> in node.js: <code>require('yoctolib-es2017/yocto_realtimeclock.js');</code>
js	<code><script type='text/javascript' src='yocto_realtimeclock.js'></script></code>
cpp	<code>#include "yocto_realtimeclock.h"</code>
m	<code>#import "yocto_realtimeclock.h"</code>
pas	<code>uses yocto_realtimeclock;</code>
vb	<code>yocto_realtimeclock.vb</code>
cs	<code>yocto_realtimeclock.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YRealTimeClock;</code>
uwp	<code>import com.yoctopuce.YoctoAPI.YRealTimeClock;</code>
py	<code>from yocto_realtimeclock import *</code>
php	<code>require_once('yocto_realtimeclock.php');</code>
ts	in HTML: <code>import { YRealTimeClock } from '../../dist/esm/yocto_realtimeclock.js';</code> in Node.js: <code>import { YRealTimeClock } from 'yoctolib-cjs/yocto_realtimeclock.js';</code>
dnp	<code>import YoctoProxyAPI.YRealTimeClockProxy</code>
cp	<code>#include "yocto_realtimeclock_proxy.h"</code>
vi	<code>YRealTimeClock.vi</code>
ml	<code>import YoctoProxyAPI.YRealTimeClockProxy</code>

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 décalage 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 décalage 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.FindRealTimeClock()

YRealTimeClock

Permet de retrouver une horloge à temps réel d'après un identifiant donné.

js	function yFindRealTimeClock (func)
cpp	YRealTimeClock* FindRealTimeClock (string func)
m	+(YRealTimeClock*) FindRealTimeClock : (NSString*) func
pas	TYRealTimeClock yFindRealTimeClock (func : string): TYRealTimeClock
vb	function FindRealTimeClock (ByVal func As String) As YRealTimeClock
cs	static YRealTimeClock FindRealTimeClock (string func)
java	static YRealTimeClock FindRealTimeClock (String func)
uwp	static YRealTimeClock FindRealTimeClock (string func)
py	FindRealTimeClock (func)
php	function FindRealTimeClock (\$func)
ts	static FindRealTimeClock (func : string): YRealTimeClock
es	static FindRealTimeClock (func)
dnp	static YRealTimeClockProxy FindRealTimeClock (string func)
cp	static YRealTimeClockProxy * FindRealTimeClock (string 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 où 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.FindRealTimeClockInContext()

YRealTimeClock

Permet de retrouver une horloge à temps réel d'après un identifiant donné dans un Contexte YAPI.

```

java static YRealTimeClock FindRealTimeClockInContext( YAPIContext yctx,
                                                         String func)
uwp static YRealTimeClock FindRealTimeClockInContext( YAPIContext yctx,
                                                         string func)
ts static FindRealTimeClockInContext( yctx: YAPIContext, func: string): YRealTimeClock
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 où 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.FirstRealTimeClock()

YRealTimeClock

Commence l'énumération des horloges à temps réel accessibles par la librairie.

js	function yFirstRealTimeClock ()
cpp	YRealTimeClock * FirstRealTimeClock ()
m	+(YRealTimeClock*) FirstRealTimeClock
pas	TYRealTimeClock yFirstRealTimeClock (): TYRealTimeClock
vb	function FirstRealTimeClock () As YRealTimeClock
cs	static YRealTimeClock FirstRealTimeClock ()
java	static YRealTimeClock FirstRealTimeClock ()
uwp	static YRealTimeClock FirstRealTimeClock ()
py	FirstRealTimeClock ()
php	function FirstRealTimeClock ()
ts	static FirstRealTimeClock (): YRealTimeClock null
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.FirstRealTimeClockInContext()

YRealTimeClock

Commence l'énumération des horloges à temps réel accessibles par la librairie.

java	static YRealTimeClock FirstRealTimeClockInContext (YAPIContext yctx)
uwp	static YRealTimeClock FirstRealTimeClockInContext (YAPIContext yctx)
ts	static FirstRealTimeClockInContext (yctx : YAPIContext): YRealTimeClock null
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).

`dn` `static new string[] GetSimilarFunctions( )`

`cp` `static vector<string> GetSimilarFunctions( )`

Chaque chaîne retournée peut être passée en argument à la méthode `YRealTimeClock.FindRealTimeClock` pour obtenir un objet permettant d'interagir 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'identifiant 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 exemple 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 exemple RELAYL01 - 123456 . r e l a y 1).

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 décalage 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 décalage 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 ()
ts	async clearCache (): Promise<void>
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 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 ()
ts	async describe (): Promise<string>
es	async describe ()

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès à 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)=RELAYL01-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.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 debugueur.

Retourne :

une chaîne de caractères décrivant l'horloge à temps réel (ex:
`Relay(MyCustomName.relay1)=RELAYL01-123456.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 ()
java	String get_advertisedValue ()
uwp	async Task<string> get_advertisedValue ()
py	get_advertisedValue ()
php	function get_advertisedValue ()
ts	async get_advertisedValue (): Promise<string>
es	async get_advertisedValue ()
dnp	string get_advertisedValue ()
cp	string 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 `YRealTimeClock.AVERTISEDVALUE_INVALID`.

realtimeclock→get_dateTime()**YRealTimeClock****realtimeclock→dateTime()**

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 ()
java	String get_dateTime ()
uwp	async Task<string> get_dateTime ()
py	get_dateTime ()
php	function get_dateTime ()
ts	async get_dateTime (): Promise<string>
es	async get_dateTime ()
dnp	string get_dateTime ()
cp	string 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 `YRealTimeClock.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 ()
ts	get_errorMessage (): string
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	<code>function get_errorType()</code>
cpp	<code>YRETCODE get_errorType()</code>
m	<code>-(YRETCODE) errorType</code>
pas	<code>YRETCODE get_errorType(): YRETCODE</code>
vb	<code>function get_errorType() As YRETCODE</code>
cs	<code>YRETCODE get_errorType()</code>
java	<code>int get_errorType()</code>
py	<code>get_errorType()</code>
php	<code>function get_errorType()</code>
ts	<code>get_errorType(): number</code>
es	<code>get_errorType()</code>

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 ()
java	String get_friendlyName ()
py	get_friendlyName ()
php	function get_friendlyName ()
ts	async get_friendlyName (): Promise<string>
es	async get_friendlyName ()
dnp	string get_friendlyName ()
cp	string 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'identifiant 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 `YRealTimeClock.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 ()
ts	async get_functionDescriptor (): Promise<string>
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\$CLASSNAME\$.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 ()
java	String get_functionId ()
py	get_functionId ()
php	function get_functionId ()
ts	async get_functionId (): Promise<string>
es	async get_functionId ()
dnp	string get_functionId ()
cp	string get_functionId ()

Par exemple `relay1`.

Retourne :

une chaîne de caractères identifiant l'horloge à temps réel (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `YRealTimeClock.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 ()
java	String get_hardwareId ()
py	get_hardwareId ()
php	function get_hardwareId ()
ts	async get_hardwareId (): Promise<string>
es	async get_hardwareId ()
dnp	string get_hardwareId ()
cp	string 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 exemple RELAYL01-123456 . relay1).

Retourne :

une chaîne de caractères identifiant l'horloge à temps réel (ex: RELAYL01-123456 . relay1)

En cas d'erreur, déclenche une exception ou retourne YRealTimeClock.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 ()
java	String get_logicalName ()
uwp	async Task<string> get_logicalName ()
py	get_logicalName ()
php	function get_logicalName ()
ts	async get_logicalName (): Promise<string>
es	async get_logicalName ()
dnp	string get_logicalName ()
cp	string 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 `YRealTimeClock.LOGICALNAME_INVALID`.

realtimeclock→get_module()**YRealTimeClock****realtimeclock→module()**

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 ()
java	YModule get_module ()
py	get_module ()
php	function get_module ()
ts	async get_module (): Promise<YModule>
es	async get_module ()
dnp	YModuleProxy get_module ()
cp	YModuleProxy * 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 ()
java	String get_serialNumber ()
uwp	async Task<string> get_serialNumber ()
py	get_serialNumber ()
php	function get_serialNumber ()
ts	async get_serialNumber (): Promise<string>
es	async get_serialNumber ()
dnp	string get_serialNumber ()
cp	string 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 YFunction.SERIALNUMBER_INVALID.

realtimeclock→get_timeSet()**YRealTimeClock****realtimeclock→timeSet()**

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 ()
java	int get_timeSet ()
uwp	async Task<int> get_timeSet ()
py	get_timeSet ()
php	function get_timeSet ()
ts	async get_timeSet (): Promise<YRealTimeClock_TimeSet>
es	async get_timeSet ()
dnp	int get_timeSet ()
cp	int get_timeSet ()
cmd	YRealTimeClock target get_timeSet

Retourne :

soit YRealTimeClock.TIMESET_FALSE, soit YRealTimeClock.TIMESET_TRUE, selon vrai si l'horloge à été mise à l'heure, sinon faux

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

realtimeclock→get_unixTime()**YRealTimeClock****realtimeclock→unixTime()**

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 ()
java	long get_unixTime ()
uwp	async Task<long> get_unixTime ()
py	get_unixTime ()
php	function get_unixTime ()
ts	async get_unixTime (): Promise<number>
es	async get_unixTime ()
dnp	long get_unixTime ()
cp	s64 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 `YRealTimeClock.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 ()
ts	async get_userdata (): Promise<object null>
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()**YRealTimeClock****realtimeclock→utcOffset()**

Retourne le nombre de secondes de décalage 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 ()
java	int get_utcOffset ()
uwp	async Task<int> get_utcOffset ()
py	get_utcOffset ()
php	function get_utcOffset ()
ts	async get_utcOffset (): Promise<number>
es	async get_utcOffset ()
dnp	int get_utcOffset ()
cp	int get_utcOffset ()
cmd	YRealTimeClock target get_utcOffset

Retourne :

un entier représentant le nombre de secondes de décalage entre l'heure courante et l'heure UTC (time zone)

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

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 ()
java	boolean isOnline ()
py	isOnline ()
php	function isOnline ()
ts	async isOnline (): Promise<boolean>
es	async isOnline ()
dnp	bool isOnline ()
cp	bool 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	<code>bool isReadOnly()</code>
m	<code>-(bool) isReadOnly</code>
pas	<code>boolean isReadOnly()</code> : boolean
vb	<code>function isReadOnly()</code> As Boolean
cs	<code>bool isReadOnly()</code>
java	<code>boolean isReadOnly()</code>
uwp	<code>async Task<bool> isReadOnly()</code>
py	<code>isReadOnly()</code>
php	<code>function isReadOnly()</code>
ts	<code>async isReadOnly()</code> : Promise<boolean>
es	<code>async isReadOnly()</code>
dnp	<code>bool isReadOnly()</code>
cp	<code>bool isReadOnly()</code>
cmd	<code>YRealTimeClock target isReadOnly</code>

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	<code>function load(msValidity)</code>
cpp	<code>YRETCODE load(int msValidity)</code>
m	<code>-(YRETCODE) load : (u64) msValidity</code>
pas	<code>YRETCODE load(msValidity: u64): YRETCODE</code>
vb	<code>function load(ByVal msValidity As Long) As YRETCODE</code>
cs	<code>YRETCODE load(ulong msValidity)</code>
java	<code>int load(long msValidity)</code>
py	<code>load(msValidity)</code>
php	<code>function load(\$msValidity)</code>
ts	<code>async load(msValidity: number): Promise<number></code>
es	<code>async load(msValidity)</code>

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 occasionnellement 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 (ByVal attrName As String) As String
cs	string loadAttribute (string attrName)
java	String loadAttribute (String attrName)
uwp	async Task<string> loadAttribute (string attrName)
py	loadAttribute (attrName)
php	function loadAttribute (\$ attrName)
ts	async loadAttribute (attrName : string): Promise<string>
es	async loadAttribute (attrName)
dnp	string loadAttribute (string attrName)
cp	string loadAttribute (string 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 occasionnellement 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 ()
java	int muteValueCallbacks ()
uwp	async Task<int> muteValueCallbacks ()
py	muteValueCallbacks ()
php	function muteValueCallbacks ()
ts	async muteValueCallbacks (): Promise<number>
es	async muteValueCallbacks ()
dnp	int muteValueCallbacks ()
cp	int 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 déclenchement de callbacks HTTP. N'oubliez pas d'appeler la méthode `saveToF lash( )` 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	<code>function nextRealTimeClock()</code>
cpp	<code>YRealTimeClock * nextRealTimeClock()</code>
m	<code>-(nullable YRealTimeClock*) nextRealTimeClock</code>
pas	<code>TYRealTimeClock nextRealTimeClock(): TYRealTimeClock</code>
vb	<code>function nextRealTimeClock() As YRealTimeClock</code>
cs	<code>YRealTimeClock nextRealTimeClock()</code>
java	<code>YRealTimeClock nextRealTimeClock()</code>
uwp	<code>YRealTimeClock nextRealTimeClock()</code>
py	<code>nextRealTimeClock()</code>
php	<code>function nextRealTimeClock()</code>
ts	<code>nextRealTimeClock(): YRealTimeClock null</code>
es	<code>nextRealTimeClock()</code>

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 _Nullable) callback
pas	LongInt registerValueCallback (callback : TYRealTimeClockValueCallback): LongInt
vb	function registerValueCallback (ByVal callback As YRealTimeClockValueCallback) 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)
ts	async registerValueCallback (callback : YRealTimeClockValueCallback null): Promise<number>
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)
java	int set_logicalName (String newval)
uwp	async Task<int> set_logicalName (string newval)
py	set_logicalName (newval)
php	function set_logicalName (\$ newval)
ts	async set_logicalName (newval : string): Promise<number>
es	async set_logicalName (newval)
dnp	int set_logicalName (string newval)
cp	int set_logicalName (string 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()**YRealTimeClock****realtimeclock→setUnixTime()**

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)
java	int set_unixTime (long newval)
uwp	async Task<int> set_unixTime (long newval)
py	set_unixTime (newval)
php	function set_unixTime (\$ newval)
ts	async set_unixTime (newval : number): Promise<number>
es	async set_unixTime (newval)
dnp	int set_unixTime (long newval)
cp	int set_unixTime (s64 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	<code>function set_userdata(data)</code>
cpp	<code>void set_userdata(void * data)</code>
m	<code>-(void) setUserData : (id) data</code>
pas	<code>set_userdata(data: Tobject)</code>
vb	<code>procedure set_userdata(ByVal data As Object)</code>
cs	<code>void set_userdata(object data)</code>
java	<code>void set_userdata(Object data)</code>
py	<code>set_userdata(data)</code>
php	<code>function set_userdata(\$data)</code>
ts	<code>async set_userdata(data: object null): Promise<void></code>
es	<code>async set_userdata(data)</code>

Cet attribut n'est 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)
java	int set_utcOffset (int newval)
uwp	async Task<int> set_utcOffset (int newval)
py	set_utcOffset (newval)
php	function set_utcOffset (\$newval)
ts	async set_utcOffset (newval : number): Promise<number>
es	async set_utcOffset (newval)
dnp	int set_utcOffset (int newval)
cp	int set_utcOffset (int newval)
cmd	YRealTimeClock target set_utcOffset newval

Le décalage 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 ()
java	int unmuteValueCallbacks ()
uwp	async Task<int> unmuteValueCallbacks ()
py	unmuteValueCallbacks ()
php	function unmuteValueCallbacks ()
ts	async unmuteValueCallbacks (): Promise<number>
es	async unmuteValueCallbacks ()
dnp	int unmuteValueCallbacks ()
cp	int 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)
ts wait_async( callback: Function, context: object)
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.

11.6. La classe YWakeUpMonitor

Interface pour interagir avec les moniteurs de réveil, disponibles par exemple dans le YoctoHub-GSM-3G-EU, le YoctoHub-GSM-3G-NA, le YoctoHub-GSM-4G et le YoctoHub-Wireless-n

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
java	import com.yoctopuce.YoctoAPI.YWakeUpMonitor;
uwp	import com.yoctopuce.YoctoAPI.YWakeUpMonitor;
py	from yocto_wakeupmonitor import *
php	require_once('yocto_wakeupmonitor.php');
ts	in HTML: import { YWakeUpMonitor } from '../../dist/esm/yocto_wakeupmonitor.js'; in Node.js: import { YWakeUpMonitor } from 'yoctolib-cjs/yocto_wakeupmonitor.js';
dnf	import YoctoProxyAPI.YWakeUpMonitorProxy
cp	#include "yocto_wakeupmonitor_proxy.h"
vi	YWakeupMonitor.vi
ml	import YoctoProxyAPI.YWakeUpMonitorProxy

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.FindWakeUpMonitor()

YWakeupMonitor

Permet de retrouver un moniteur de réveil d'après un identifiant donné.

js	function yFindWakeUpMonitor (func)
cpp	YWakeupMonitor* FindWakeUpMonitor (string func)
m	+(YWakeupMonitor*) FindWakeUpMonitor : (NSString*) func
pas	TYWakeUpMonitor yFindWakeUpMonitor (func : string): TYWakeUpMonitor
vb	function FindWakeUpMonitor (ByVal func As String) As YWakeupMonitor
cs	static YWakeupMonitor FindWakeUpMonitor (string func)
java	static YWakeupMonitor FindWakeUpMonitor (String func)
uwp	static YWakeupMonitor FindWakeUpMonitor (string func)
py	FindWakeUpMonitor (func)
php	function FindWakeUpMonitor (\$func)
ts	static FindWakeUpMonitor (func : string): YWakeupMonitor
es	static FindWakeUpMonitor (func)
dnp	static YWakeupMonitorProxy FindWakeUpMonitor (string func)
cp	static YWakeupMonitorProxy * FindWakeUpMonitor (string 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 où 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 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.FindWakeUpMonitorInContext()

YWakeupMonitor

Permet de retrouver un moniteur de réveil d'après un identifiant donné dans un Contexte YAPI.

```

java static YWakeUpMonitor FindWakeUpMonitorInContext( YAPIContext yctx,
                                                         String func)

uwp static YWakeUpMonitor FindWakeUpMonitorInContext( YAPIContext yctx,
                                                         string func)

ts static FindWakeUpMonitorInContext( yctx: YAPIContext, func: string): YWakeUpMonitor

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 où 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.FirstWakeUpMonitor()

YWakeupMonitor

Commence l'énumération des moniteurs de réveil accessibles par la librairie.

js	function yFirstWakeUpMonitor ()
cpp	YWakeupMonitor * FirstWakeUpMonitor ()
m	+(YWakeupMonitor*) FirstWakeUpMonitor
pas	TYWakeupMonitor yFirstWakeUpMonitor (): TYWakeupMonitor
vb	function FirstWakeUpMonitor () As YWakeupMonitor
cs	static YWakeupMonitor FirstWakeUpMonitor ()
java	static YWakeupMonitor FirstWakeUpMonitor ()
uwp	static YWakeupMonitor FirstWakeUpMonitor ()
py	FirstWakeUpMonitor ()
php	function FirstWakeUpMonitor ()
ts	static FirstWakeUpMonitor (): YWakeupMonitor null
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.FirstWakeUpMonitorInContext()

YWakeupMonitor

Commence l'énumération des moniteurs de réveil accessibles par la librairie.

java	static YWakeUpMonitor FirstWakeUpMonitorInContext (YAPIContext yctx)
uwp	static YWakeUpMonitor FirstWakeUpMonitorInContext (YAPIContext yctx)
ts	static FirstWakeUpMonitorInContext (yctx : YAPIContext): YWakeUpMonitor null
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	<code>static new string[] GetSimilarFunctions()</code>
-----	--

cp	<code>static vector<string> GetSimilarFunctions()</code>
----	--

Chaque chaîne retournée peut être passée en argument à la méthode `YWakeUpMonitor.FindWakeUpMonitor` pour obtenir un objet permettant d'interagir 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'identifiant 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 exemple 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 exemple RELAYL01 - 123456 . r e l a y 1).

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).

`dnf` `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 ()
ts	async clearCache (): Promise<void>
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 ()
ts	async describe (): Promise<string>
es	async describe ()

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès à 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 méthode va retourner `Relay(MyCustomName.relay1)=RELAYL01-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

Retourne :

une chaîne de caractères décrivant le moniteur de réveil (ex:
`Relay(MyCustomName.relay1)=RELAYL01-123456.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 ()
java	String get_advertisedValue ()
uwp	async Task<string> get_advertisedValue ()
py	get_advertisedValue ()
php	function get_advertisedValue ()
ts	async get_advertisedValue (): Promise<string>
es	async get_advertisedValue ()
dnp	string get_advertisedValue ()
cp	string 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 `YWakeUpMonitor.AVERTISEDVALUE_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 ()
ts	get_errorMessage (): string
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	<code>function get_errorType()</code>
cpp	<code>YRETCODE get_errorType()</code>
m	<code>-(YRETCODE) errorType</code>
pas	<code>YRETCODE get_errorType(): YRETCODE</code>
vb	<code>function get_errorType() As YRETCODE</code>
cs	<code>YRETCODE get_errorType()</code>
java	<code>int get_errorType()</code>
py	<code>get_errorType()</code>
php	<code>function get_errorType()</code>
ts	<code>get_errorType(): number</code>
es	<code>get_errorType()</code>

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 ()
java	String get_friendlyName ()
py	get_friendlyName ()
php	function get_friendlyName ()
ts	async get_friendlyName (): Promise<string>
es	async get_friendlyName ()
dnp	string get_friendlyName ()
cp	string 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'identifiant 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 `YWakeupMonitor.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 ()
ts	async get_functionDescriptor (): Promise<string>
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\$CLASSNAME\$.FUNCTIONDESCRIPTOR_INVALID

wakeupmonitor→get_functionId()**YWakeUpMonitor****wakeupmonitor→functionId()**

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 ()
java	String get_functionId ()
py	get_functionId ()
php	function get_functionId ()
ts	async get_functionId (): Promise<string>
es	async get_functionId ()
dnp	string get_functionId ()
cp	string get_functionId ()

Par exemple `relay1`.

Retourne :

une chaîne de caractères identifiant le moniteur de réveil (ex: `relay1`)

En cas d'erreur, déclenche une exception ou retourne `YWakeupMonitor.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 ()
java	String get_hardwareId ()
py	get_hardwareId ()
php	function get_hardwareId ()
ts	async get_hardwareId (): Promise<string>
es	async get_hardwareId ()
dnp	string get_hardwareId ()
cp	string get_hardwareId ()

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du moniteur de réveil (par exemple RELAYL01-123456 . relay1).

Retourne :

une chaîne de caractères identifiant le moniteur de réveil (ex: RELAYL01-123456 . relay1)

En cas d'erreur, déclenche une exception ou retourne `YWakeUpMonitor.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 ()
java	String get_logicalName ()
uwp	async Task<string> get_logicalName ()
py	get_logicalName ()
php	function get_logicalName ()
ts	async get_logicalName (): Promise<string>
es	async get_logicalName ()
dnp	string get_logicalName ()
cp	string 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 `YWakeupMonitor.LOGICALNAME_INVALID`.

wakeupmonitor→get_module()**YWakeUpMonitor****wakeupmonitor→module()**

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 ()
java	YModule get_module ()
py	get_module ()
php	function get_module ()
ts	async get_module (): Promise<YModule>
es	async get_module ()
dnp	YModuleProxy get_module ()
cp	YModuleProxy * 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 ()
java	long get_nextWakeUp ()
uwp	async Task<long> get_nextWakeUp ()
py	get_nextWakeUp ()
php	function get_nextWakeUp ()
ts	async get_nextWakeUp (): Promise<number>
es	async get_nextWakeUp ()
dnp	long get_nextWakeUp ()
cp	s64 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 `YWakeUpMonitor.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 ()
java	int get_powerDuration ()
uwp	async Task<int> get_powerDuration ()
py	get_powerDuration ()
php	function get_powerDuration ()
ts	async get_powerDuration (): Promise<number>
es	async get_powerDuration ()
dnp	int get_powerDuration ()
cp	int 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 YWakeupMonitor.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 ()
java	String get_serialNumber ()
uwp	async Task<string> get_serialNumber ()
py	get_serialNumber ()
php	function get_serialNumber ()
ts	async get_serialNumber (): Promise<string>
es	async get_serialNumber ()
dnp	string get_serialNumber ()
cp	string 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 YFunction.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 ()
java	int get_sleepCountdown ()
uwp	async Task<int> get_sleepCountdown ()
py	get_sleepCountdown ()
php	function get_sleepCountdown ()
ts	async get_sleepCountdown (): Promise<number>
es	async get_sleepCountdown ()
dnp	int get_sleepCountdown ()
cp	int 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 `YWakeUpMonitor.SLEEP_COUNTDOWN_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 ()
ts	async get_userdata (): Promise<object null>
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 ()
java	int get_wakeUpReason ()
uwp	async Task<int> get_wakeUpReason ()
py	get_wakeUpReason ()
php	function get_wakeUpReason ()
ts	async get_wakeUpReason (): Promise<YWakeUpMonitor_WakeUpReason>
es	async get_wakeUpReason ()
dnp	int get_wakeUpReason ()
cp	int get_wakeUpReason ()
cmd	YWakeUpMonitor target get_wakeUpReason

Retourne :

une valeur parmi YWakeUpMonitor.WAKEUPREASON_USBPOWER,
 YWakeUpMonitor.WAKEUPREASON_EXTPOWER,
 YWakeUpMonitor.WAKEUPREASON_ENDOFSLEEP,
 YWakeUpMonitor.WAKEUPREASON_EXTSIG1,
 YWakeUpMonitor.WAKEUPREASON_SCHEDULE1 et
 YWakeUpMonitor.WAKEUPREASON_SCHEDULE2

En cas d'erreur, déclenche une exception ou retourne
 YWakeUpMonitor.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 ()
java	int get_wakeUpState ()
uwp	async Task<int> get_wakeUpState ()
py	get_wakeUpState ()
php	function get_wakeUpState ()
ts	async get_wakeUpState (): Promise<YWakeUpMonitor_WakeUpState>
es	async get_wakeUpState ()
dnp	int get_wakeUpState ()
cp	int get_wakeUpState ()
cmd	YWakeUpMonitor target get_wakeUpState

Retourne :

soit YWakeUpMonitor.WAKEUPSTATE_SLEEPING, soit
 YWakeUpMonitor.WAKEUPSTATE_AWAKE

En cas d'erreur, déclenche une exception ou retourne
 YWakeUpMonitor.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 ()
java	boolean isOnline ()
py	isOnline ()
php	function isOnline ()
ts	async isOnline (): Promise<boolean>
es	async isOnline ()
dnp	bool isOnline ()
cp	bool 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 ()
java	boolean isReadOnly ()
uwp	async Task<bool> isReadOnly ()
py	isReadOnly ()
php	function isReadOnly ()
ts	async isReadOnly (): Promise<boolean>
es	async isReadOnly ()
dnp	bool isReadOnly ()
cp	bool 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	<code>function load(msValidity)</code>
cpp	<code>YRETCODE load(int msValidity)</code>
m	<code>-(YRETCODE) load : (u64) msValidity</code>
pas	<code>YRETCODE load(msValidity: u64): YRETCODE</code>
vb	<code>function load(ByVal msValidity As Long) As YRETCODE</code>
cs	<code>YRETCODE load(ulong msValidity)</code>
java	<code>int load(long msValidity)</code>
py	<code>load(msValidity)</code>
php	<code>function load(\$msValidity)</code>
ts	<code>async load(msValidity: number): Promise<number></code>
es	<code>async load(msValidity)</code>

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 occasionnellement 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 (ByVal attrName As String) As String
cs	string loadAttribute (string attrName)
java	String loadAttribute (String attrName)
uwp	async Task<string> loadAttribute (string attrName)
py	loadAttribute (attrName)
php	function loadAttribute (\$ attrName)
ts	async loadAttribute (attrName : string): Promise<string>
es	async loadAttribute (attrName)
dnp	string loadAttribute (string attrName)
cp	string loadAttribute (string 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 occasionnellement 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 ()
java	int muteValueCallbacks ()
uwp	async Task<int> muteValueCallbacks ()
py	muteValueCallbacks ()
php	function muteValueCallbacks ()
ts	async muteValueCallbacks (): Promise<number>
es	async muteValueCallbacks ()
dnp	int muteValueCallbacks ()
cp	int 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 déclenchement de callbacks HTTP. N'oubliez pas d'appeler la méthode `saveToF lash( )` 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	<code>function nextWakeUpMonitor()</code>
cpp	<code>YWakeupMonitor * nextWakeUpMonitor()</code>
m	<code>-(nullable YWakeupMonitor*) nextWakeUpMonitor</code>
pas	<code>TYWakeupMonitor nextWakeUpMonitor(): TYWakeupMonitor</code>
vb	<code>function nextWakeUpMonitor() As YWakeupMonitor</code>
cs	<code>YWakeupMonitor nextWakeUpMonitor()</code>
java	<code>YWakeupMonitor nextWakeUpMonitor()</code>
uwp	<code>YWakeupMonitor nextWakeUpMonitor()</code>
py	<code>nextWakeUpMonitor()</code>
php	<code>function nextWakeUpMonitor()</code>
ts	<code>nextWakeUpMonitor(): YWakeupMonitor null</code>
es	<code>nextWakeUpMonitor()</code>

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 _Nullable) callback
pas	LongInt registerValueCallback (callback : TYWakeUpMonitorValueCallback): LongInt
vb	function registerValueCallback (ByVal callback As YWakeUpMonitorValueCallback) 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)
ts	async registerValueCallback (callback : YWakeUpMonitorValueCallback null): Promise<number>
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 ()
java	int resetSleepCountDown ()
uwp	async Task<int> resetSleepCountDown ()
py	resetSleepCountDown ()
php	function resetSleepCountDown ()
ts	async resetSleepCountDown (): Promise<number>
es	async resetSleepCountDown ()
dnp	int resetSleepCountDown ()
cp	int 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()****YWakeUpMonitor****wakeupmonitor**→**setLogicalName()**

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)
java	int set_logicalName (String newval)
uwp	async Task<int> set_logicalName (string newval)
py	set_logicalName (newval)
php	function set_logicalName (\$newval)
ts	async set_logicalName (newval : string): Promise<number>
es	async set_logicalName (newval)
dnp	int set_logicalName (string newval)
cp	int set_logicalName (string 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)
java	int set_nextWakeUp (long newval)
uwp	async Task<int> set_nextWakeUp (long newval)
py	set_nextWakeUp (newval)
php	function set_nextWakeUp (\$ newval)
ts	async set_nextWakeUp (newval : number): Promise<number>
es	async set_nextWakeUp (newval)
dnp	int set_nextWakeUp (long newval)
cp	int set_nextWakeUp (s64 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.

wakeuptime→set_powerDuration()

YWakeUpMonitor

wakeuptime→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)
java	int set_powerDuration(int newval)
uwp	async Task<int> set_powerDuration(int newval)
py	set_powerDuration(newval)
php	function set_powerDuration(\$newval)
ts	async set_powerDuration(newval: number): Promise<number>
es	async set_powerDuration(newval)
dnp	int set_powerDuration(int newval)
cp	int set_powerDuration(int 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)
java	int set_sleepCountdown (int newval)
uwp	async Task<int> set_sleepCountdown (int newval)
py	set_sleepCountdown (newval)
php	function set_sleepCountdown (\$newval)
ts	async set_sleepCountdown (newval : number): Promise<number>
es	async set_sleepCountdown (newval)
dnp	int set_sleepCountdown (int newval)
cp	int set_sleepCountdown (int 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()**YWakeUpMonitor****wakeupmonitor→setUserData()**

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

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)
ts	async set_userData (data : object null): Promise<void>
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 (ByVal secBeforeSleep As Integer) As Integer
cs	int sleep (int secBeforeSleep)
java	int sleep (int secBeforeSleep)
uwp	async Task<int> sleep (int secBeforeSleep)
py	sleep (secBeforeSleep)
php	function sleep (\$ secBeforeSleep)
ts	async sleep (secBeforeSleep : number): Promise<number>
es	async sleep (secBeforeSleep)
dnp	int sleep (int secBeforeSleep)
cp	int sleep (int 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 (ByVal secUntilWakeUp As Integer, ByVal secBeforeSleep As Integer) As Integer
cs	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)
ts	async sleepFor (secUntilWakeUp : number, secBeforeSleep : number): Promise<number>
es	async sleepFor (secUntilWakeUp , secBeforeSleep)
dnp	int sleepFor (int secUntilWakeUp , int secBeforeSleep)
cp	int sleepFor (int secUntilWakeUp , int 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 (ByVal wakeUpTime As Integer, ByVal secBeforeSleep As Integer) As Integer
cs	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)
ts	async sleepUntil (wakeUpTime : number, secBeforeSleep : number): Promise<number>
es	async sleepUntil (wakeUpTime , secBeforeSleep)
dnp	int sleepUntil (int wakeUpTime , int secBeforeSleep)
cp	int sleepUntil (int wakeUpTime , int 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 ()
java	int unmuteValueCallbacks ()
uwp	async Task<int> unmuteValueCallbacks ()
py	unmuteValueCallbacks ()
php	function unmuteValueCallbacks ()
ts	async unmuteValueCallbacks (): Promise<number>
es	async unmuteValueCallbacks ()
dnp	int unmuteValueCallbacks ()
cp	int 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	<code>function wait_async(callback, context)</code>
ts	<code>wait_async(callback: Function, context: object)</code>
es	<code>wait_async(callback, context)</code>

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 ()
java	int wakeUp ()
uwp	async Task<int> wakeUp ()
py	wakeUp ()
php	function wakeUp ()
ts	async wakeUp (): Promise<number>
es	async wakeUp ()
dnp	int wakeUp ()
cp	int wakeUp ()
cmd	YWakeUpMonitor target wakeUp

11.7. La classe YWakeUpSchedule

Interface pour interagir avec les réveils agendés, disponibles par exemple dans le YoctoHub-GSM-3G-EU, le YoctoHub-GSM-3G-NA, le YoctoHub-GSM-4G et le YoctoHub-Wireless-n

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: <code><script src="../../lib/yocto_wakeupschedule.js"></script></code> in node.js: <code>require('yoctolib-es2017/yocto_wakeupschedule.js');</code>
js	<code><script type='text/javascript' src='yocto_wakeupschedule.js'></script></code>
cpp	<code>#include "yocto_wakeupschedule.h"</code>
m	<code>#import "yocto_wakeupschedule.h"</code>
pas	<code>uses yocto_wakeupschedule;</code>
vb	<code>yocto_wakeupschedule.vb</code>
cs	<code>yocto_wakeupschedule.cs</code>
java	<code>import com.yoctopuce.YoctoAPI.YWakeUpSchedule;</code>
uwp	<code>import com.yoctopuce.YoctoAPI.YWakeUpSchedule;</code>
py	<code>from yocto_wakeupschedule import *</code>
php	<code>require_once('yocto_wakeupschedule.php');</code>
ts	in HTML: <code>import { YWakeUpSchedule } from '../../dist/esm/yocto_wakeupschedule.js';</code> in Node.js: <code>import { YWakeUpSchedule } from 'yoctolib-cjs/yocto_wakeupschedule.js';</code>
dnf	<code>import YoctoProxyAPI.YWakeUpScheduleProxy</code>
cp	<code>#include "yocto_wakeupschedule_proxy.h"</code>
vi	<code>YWakeupSchedule.vi</code>
ml	<code>import YoctoProxyAPI.YWakeUpScheduleProxy</code>

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'intervall 00-29 de chaque heures où le réveil est actif.

wakeupschedule→MinutesB *[modifiable]*

S minutes de l'intervall 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 occurrence 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'intervall 00-29 de chaque heures où le réveil est actif.

wakeupschedule→get_minutesB()

Retourne les minutes de l'intervall 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 occurrence 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'intervall 00-29 où un réveil doit avoir lieu.

wakeupschedule→set_minutesB(newval)

Modifie les minutes de l'intervall 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.FindWakeUpSchedule()

YWakeupSchedule

Permet de retrouver un réveil agendé d'après un identifiant donné.

js	function yFindWakeUpSchedule (func)
cpp	YWakeupSchedule* FindWakeUpSchedule (string func)
m	+(YWakeupSchedule*) FindWakeUpSchedule : (NSString*) func
pas	TYWakeUpSchedule yFindWakeUpSchedule (func : string): TYWakeUpSchedule
vb	function FindWakeUpSchedule (ByVal func As String) As YWakeupSchedule
cs	static YWakeupSchedule FindWakeUpSchedule (string func)
java	static YWakeupSchedule FindWakeUpSchedule (String func)
uwp	static YWakeupSchedule FindWakeUpSchedule (string func)
py	FindWakeUpSchedule (func)
php	function FindWakeUpSchedule (\$func)
ts	static FindWakeUpSchedule (func : string): YWakeupSchedule
es	static FindWakeUpSchedule (func)
dnp	static YWakeupScheduleProxy FindWakeUpSchedule (string func)
cp	static YWakeupScheduleProxy * FindWakeUpSchedule (string 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 où 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.FindWakeUpScheduleInContext()

YWakeUpSchedule

Permet de retrouver un réveil agendé d'après un identifiant donné dans un Contexte YAPI.

```

java static YWakeUpSchedule FindWakeUpScheduleInContext( YAPIContext yctx,
                                                         String func)
uwp static YWakeUpSchedule FindWakeUpScheduleInContext( YAPIContext yctx,
                                                         string func)
ts static FindWakeUpScheduleInContext( yctx: YAPIContext, func: string): YWakeUpSchedule
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 où 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.FirstWakeUpSchedule()

YWakeupSchedule

Commence l'énumération des réveils agendés accessibles par la librairie.

js	function yFirstWakeUpSchedule ()
cpp	YWakeupSchedule * FirstWakeUpSchedule ()
m	+(YWakeupSchedule*) FirstWakeUpSchedule
pas	TYWakeupSchedule yFirstWakeUpSchedule (): TYWakeupSchedule
vb	function FirstWakeUpSchedule () As YWakeupSchedule
cs	static YWakeupSchedule FirstWakeUpSchedule ()
java	static YWakeupSchedule FirstWakeUpSchedule ()
uwp	static YWakeupSchedule FirstWakeUpSchedule ()
py	FirstWakeUpSchedule ()
php	function FirstWakeUpSchedule ()
ts	static FirstWakeUpSchedule (): YWakeupSchedule null
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.FirstWakeUpScheduleInContext()

YWakeUpSchedule

Commence l'énumération des réveils agendés accessibles par la librairie.

java	static YWakeUpSchedule FirstWakeUpScheduleInContext (YAPIContext yctx)
uwp	static YWakeUpSchedule FirstWakeUpScheduleInContext (YAPIContext yctx)
ts	static FirstWakeUpScheduleInContext (yctx : YAPIContext): YWakeUpSchedule null
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.GetSimilarFunctions()

YWakeupSchedule

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

`dnv` `static new string[] GetSimilarFunctions( )`

`cp` `static vector<string> GetSimilarFunctions( )`

Chaque chaîne retournée peut être passée en argument à la méthode `YWakeupSchedule.FindWakeUpSchedule` pour obtenir un objet permettant d'interagir 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'identifiant matériel de la fonction (par exemple: `MyCustomName.relay1`)

wakeupschedule→FunctionId

YWakeUpSchedule

Identifiant matériel du réveil agendé, sans référence au module.

dnp `string` **FunctionId**

Par exemple 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 exemple RELAYL01 - 123456 . r e l a y 1).

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.

dnf **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'intervall 30-59 de chaque heure où le réveil est actif.

dnp **int MinutesB**

Modifiable. Modifie les minutes de l'intervall 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 occurrence 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 ()
ts	async clearCache (): Promise<void>
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 ()
ts	async describe (): Promise<string>
es	async describe ()

Plus précisément, TYPE correspond au type de fonction, NAME correspond au nom utilisé lors du premier accès à 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 méthode va retourner `Relay(MyCustomName.relay1)=RELAYL01-123456.relay1` si le module est déjà connecté ou `Relay(BadCustomName.relay1)=unresolved` si le module n'est pas déjà connecté. Cette méthode ne déclenche aucune transaction USB ou TCP et peut donc être utilisé dans un débogueur.

Retourne :

une chaîne de caractères décrivant le réveil agendé (ex:
`Relay(MyCustomName.relay1)=RELAYL01-123456.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 ()
java	String get_advertisedValue ()
uwp	async Task<string> get_advertisedValue ()
py	get_advertisedValue ()
php	function get_advertisedValue ()
ts	async get_advertisedValue (): Promise<string>
es	async get_advertisedValue ()
dnp	string get_advertisedValue ()
cp	string 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 `YWakeUpSchedule.AVERTISEDVALUE_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 ()
ts	get_errorMessage (): string
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 ()
ts	get_errorType (): number
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 ()
java	String get_friendlyName ()
py	get_friendlyName ()
php	function get_friendlyName ()
ts	async get_friendlyName (): Promise<string>
es	async get_friendlyName ()
dnp	string get_friendlyName ()
cp	string 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'identifiant 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 `YWakeUpSchedule.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 ()
ts	async get_functionDescriptor (): Promise<string>
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\$CLASSNAME\$.FUNCTIONDESCRIPTOR_INVALID

wakeupschedule→get_functionId()**YWakeUpSchedule****wakeupschedule→functionId()**

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 ()
java	String get_functionId ()
py	get_functionId ()
php	function get_functionId ()
ts	async get_functionId (): Promise<string>
es	async get_functionId ()
dnp	string get_functionId ()
cp	string get_functionId ()

Par exemple `relay1`.

Retourne :

une chaîne de caractères identifiant le réveil agendé (ex: `relay1`)

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

wakeupschedule→get_hardwareId()**YWakeUpSchedule****wakeupschedule→hardwareId()**

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 ()
java	String get_hardwareId ()
py	get_hardwareId ()
php	function get_hardwareId ()
ts	async get_hardwareId (): Promise<string>
es	async get_hardwareId ()
dnp	string get_hardwareId ()
cp	string get_hardwareId ()

L'identifiant unique est composé du numéro de série du module et de l'identifiant matériel du réveil agendé (par exemple RELAYL01-123456 . relay1).

Retourne :

une chaîne de caractères identifiant le réveil agendé (ex: RELAYL01-123456 . relay1)

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

wakeupschedule→get_hours()**YWakeUpSchedule****wakeupschedule→hours()**

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 ()
java	int get_hours ()
uwp	async Task<int> get_hours ()
py	get_hours ()
php	function get_hours ()
ts	async get_hours (): Promise<number>
es	async get_hours ()
dnp	int get_hours ()
cp	int 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 YWakeUpSchedule.HOURS_INVALID.

wakeupschedule→get_logicalName()**YWakeUpSchedule****wakeupschedule→logicalName()**

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 ()
java	String get_logicalName ()
uwp	async Task<string> get_logicalName ()
py	get_logicalName ()
php	function get_logicalName ()
ts	async get_logicalName (): Promise<string>
es	async get_logicalName ()
dnp	string get_logicalName ()
cp	string 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 `YWakeUpSchedule.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 ()
java	long get_minutes ()
uwp	async Task<long> get_minutes ()
py	get_minutes ()
php	function get_minutes ()
ts	async get_minutes (): Promise<number>
es	async get_minutes ()
dnp	long get_minutes ()
cp	s64 get_minutes ()
cmd	YWakeUpSchedule target get_minutes

wakeupschedule→get_minutesA()**YWakeUpSchedule****wakeupschedule→minutesA()**

Retourne les minutes de l'intervall 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 ()
java	int get_minutesA ()
uwp	async Task<int> get_minutesA ()
py	get_minutesA ()
php	function get_minutesA ()
ts	async get_minutesA (): Promise<number>
es	async get_minutesA ()
dnp	int get_minutesA ()
cp	int get_minutesA ()
cmd	YWakeUpSchedule target get_minutesA

Retourne :

un entier représentant les minutes de l'intervall 00-29 de chaque heures où le réveil est actif

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

wakeupschedule→get_minutesB()**YWakeUpSchedule****wakeupschedule→minutesB()**

Retourne les minutes de l'intervall 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 ()
java	int get_minutesB ()
uwp	async Task<int> get_minutesB ()
py	get_minutesB ()
php	function get_minutesB ()
ts	async get_minutesB (): Promise<number>
es	async get_minutesB ()
dnp	int get_minutesB ()
cp	int get_minutesB ()
cmd	YWakeUpSchedule target get_minutesB

Retourne :

un entier représentant les minutes de l'intervall 30-59 de chaque heure où le réveil est actif

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

wakeupschedule→**get_module()****YWakeUpSchedule****wakeupschedule**→**module()**

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 ()
java	YModule get_module ()
py	get_module ()
php	function get_module ()
ts	async get_module (): Promise<YModule>
es	async get_module ()
dnp	YModuleProxy get_module ()
cp	YModuleProxy * 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()

YWakeUpSchedule

wakeupschedule→monthDays()

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 ()
java	int get_monthDays ()
uwp	async Task<int> get_monthDays ()
py	get_monthDays ()
php	function get_monthDays ()
ts	async get_monthDays (): Promise<number>
es	async get_monthDays ()
dnp	int get_monthDays ()
cp	int 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 `YWakeUpSchedule.MONTHDAYS_INVALID`.

wakeupschedule→get_months()**YWakeUpSchedule****wakeupschedule→months()**

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 ()
java	int get_months ()
uwp	async Task<int> get_months ()
py	get_months ()
php	function get_months ()
ts	async get_months (): Promise<number>
es	async get_months ()
dnp	int get_months ()
cp	int 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 YWakeUpSchedule.MONTHS_INVALID.

wakeupschedule→get_nextOccurence()

YWakeUpSchedule

wakeupschedule→nextOccurence()

Retourne la date/heure de la prochaine occurrence 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 ()
java	long get_nextOccurence ()
uwp	async Task<long> get_nextOccurence ()
py	get_nextOccurence ()
php	function get_nextOccurence ()
ts	async get_nextOccurence (): Promise<number>
es	async get_nextOccurence ()
dnp	long get_nextOccurence ()
cp	s64 get_nextOccurence ()
cmd	YWakeUpSchedule target get_nextOccurence

Retourne :

un entier représentant la date/heure de la prochaine occurrence de réveil

En cas d'erreur, déclenche une exception ou retourne `YWakeUpSchedule.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 ()
java	String get_serialNumber ()
uwp	async Task<string> get_serialNumber ()
py	get_serialNumber ()
php	function get_serialNumber ()
ts	async get_serialNumber (): Promise<string>
es	async get_serialNumber ()
dnp	string get_serialNumber ()
cp	string 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 YFunction.SERIALNUMBER_INVALID.

wakeupschedule→get_userdata()**YWakeUpSchedule****wakeupschedule→userdata()**

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

js	<code>function get_userdata()</code>
cpp	<code>void * get_userdata()</code>
m	<code>-(id) userData</code>
pas	<code>Tobject get_userdata(): Tobject</code>
vb	<code>function get_userdata() As Object</code>
cs	<code>object get_userdata()</code>
java	<code>Object get_userdata()</code>
py	<code>get_userdata()</code>
php	<code>function get_userdata()</code>
ts	<code>async get_userdata(): Promise<object null></code>
es	<code>async get_userdata()</code>

Cet attribut n'est 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()**YWakeUpSchedule****wakeupschedule→weekDays()**

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 ()
java	int get_weekDays ()
uwp	async Task<int> get_weekDays ()
py	get_weekDays ()
php	function get_weekDays ()
ts	async get_weekDays (): Promise<number>
es	async get_weekDays ()
dnp	int get_weekDays ()
cp	int 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 `YWakeUpSchedule.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 ()
java	boolean isOnline ()
py	isOnline ()
php	function isOnline ()
ts	async isOnline (): Promise<boolean>
es	async isOnline ()
dnp	bool isOnline ()
cp	bool 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	<code>bool isReadOnly()</code>
m	<code>-(bool) isReadOnly</code>
pas	<code>boolean isReadOnly(): boolean</code>
vb	<code>function isReadOnly() As Boolean</code>
cs	<code>bool isReadOnly()</code>
java	<code>boolean isReadOnly()</code>
uwp	<code>async Task<bool> isReadOnly()</code>
py	<code>isReadOnly()</code>
php	<code>function isReadOnly()</code>
ts	<code>async isReadOnly(): Promise<boolean></code>
es	<code>async isReadOnly()</code>
dnp	<code>bool isReadOnly()</code>
cp	<code>bool isReadOnly()</code>
cmd	<code>YWakeUpSchedule target isReadOnly</code>

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)
ts	async load (msValidity : number): Promise<number>
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 occasionnellement 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 (ByVal attrName As String) As String
cs	string loadAttribute (string attrName)
java	String loadAttribute (String attrName)
uwp	async Task<string> loadAttribute (string attrName)
py	loadAttribute (attrName)
php	function loadAttribute (\$ attrName)
ts	async loadAttribute (attrName : string): Promise<string>
es	async loadAttribute (attrName)
dnp	string loadAttribute (string attrName)
cp	string loadAttribute (string 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 occasionnellement 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 ()
java	int muteValueCallbacks ()
uwp	async Task<int> muteValueCallbacks ()
py	muteValueCallbacks ()
php	function muteValueCallbacks ()
ts	async muteValueCallbacks (): Promise<number>
es	async muteValueCallbacks ()
dnp	int muteValueCallbacks ()
cp	int 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 déclenchement 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	-(nullable 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()
ts	nextWakeUpSchedule() : YWakeUpSchedule null
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 _Nullable) callback
pas	LongInt registerValueCallback (callback : TYWakeUpScheduleValueCallback): LongInt
vb	function registerValueCallback (ByVal callback As YWakeUpScheduleValueCallback) 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)
ts	async registerValueCallback (callback : YWakeUpScheduleValueCallback null): Promise<number>
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()

YWakeUpSchedule

wakeupschedule→setHours()

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)
java	int set_hours (int newval)
uwp	async Task<int> set_hours (int newval)
py	set_hours (newval)
php	function set_hours (\$newval)
ts	async set_hours (newval : number): Promise<number>
es	async set_hours (newval)
dnp	int set_hours (int newval)
cp	int set_hours (int 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()

YWakeUpSchedule

wakeupschedule→setLogicalName()

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)
java	int set_logicalName(String newval)
uwp	async Task<int> set_logicalName(string newval)
py	set_logicalName(newval)
php	function set_logicalName(\$newval)
ts	async set_logicalName(newval: string): Promise<number>
es	async set_logicalName(newval)
dnp	int set_logicalName(string newval)
cp	int set_logicalName(string 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 (ByVal bitmap As Long) As Integer
cs	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)
ts	async set_minutes (bitmap : number): Promise<number>
es	async set_minutes (bitmap)
dnp	int set_minutes (long bitmap)
cp	int set_minutes (s64 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'intervall 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)
java	int set_minutesA (int newval)
uwp	async Task<int> set_minutesA (int newval)
py	set_minutesA (newval)
php	function set_minutesA (\$newval)
ts	async set_minutesA (newval : number): Promise<number>
es	async set_minutesA (newval)
dnp	int set_minutesA (int newval)
cp	int set_minutesA (int 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'intervall 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)
java	int set_minutesB (int newval)
uwp	async Task<int> set_minutesB (int newval)
py	set_minutesB (newval)
php	function set_minutesB (\$ newval)
ts	async set_minutesB (newval : number): Promise<number>
es	async set_minutesB (newval)
dnp	int set_minutesB (int newval)
cp	int set_minutesB (int 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)
java	int set_monthDays(int newval)
uwp	async Task<int> set_monthDays(int newval)
py	set_monthDays(newval)
php	function set_monthDays(\$newval)
ts	async set_monthDays(newval: number): Promise<number>
es	async set_monthDays(newval)
dnp	int set_monthDays(int newval)
cp	int set_monthDays(int 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()

YWakeUpSchedule

wakeupschedule→setMonths()

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)
java	int set_months (int newval)
uwp	async Task<int> set_months (int newval)
py	set_months (newval)
php	function set_months (\$newval)
ts	async set_months (newval : number): Promise<number>
es	async set_months (newval)
dnp	int set_months (int newval)
cp	int set_months (int 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()**YWakeUpSchedule****wakeupschedule→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)
ts	async set_userdata (data : object null): Promise<void>
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)
java	int set_weekDays (int newval)
uwp	async Task<int> set_weekDays (int newval)
py	set_weekDays (newval)
php	function set_weekDays (\$newval)
ts	async set_weekDays (newval : number): Promise<number>
es	async set_weekDays (newval)
dnp	int set_weekDays (int newval)
cp	int set_weekDays (int 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 ()
java	int unmuteValueCallbacks ()
uwp	async Task<int> unmuteValueCallbacks ()
py	unmuteValueCallbacks ()
php	function unmuteValueCallbacks ()
ts	async unmuteValueCallbacks (): Promise<number>
es	async unmuteValueCallbacks ()
dnp	int unmuteValueCallbacks ()
cp	int 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	<code>function wait_async(callback, context)</code>
ts	<code>wait_async(callback: Function, context: object)</code>
es	<code>wait_async(callback, context)</code>

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-qu'il est à la fonction de callback

Retourne :

rien du tout.

12. Problèmes courants

12.1. 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.

12.2. 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 exécutable 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 éviteront 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"
```

12.3. Plateformes ARM: HF et EL

Sur ARM il existe deux grandes familles d'exécutables: 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 ArmHF 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 appartient, essayez simplement de lancer un exécutable de chaque famille.

12.4. 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.

12.5. Module alimenté mais invisible pour l'OS

Si votre YoctoHub-Wireless-SR 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.

12.6. 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>

12.7. Déconnexions, comportement erratique

Si votre YoctoHub-Wireless-SR se comporte de manière erratique et/ou se déconnecte du bus USB sans raison apparente, vérifiez qu'il est alimenté correctement. Évitez les câbles d'une longueur supérieure à 2 mètres. Au besoin, intercalez un hub USB alimenté ^{5 6}.

12.8. RegisterHub d'un VirtualHub déconnecte le précédent

Si lorsque vous faire un `YAPI.RegisterHub` d'un VirtualHub la connexion avec un autre virtualHub précédemment enregistré tombe, vérifiez que les machines qui hébergent ces VirtualHubs ont bien un *hostname* différent. Ce cas de figure est très courant avec les machines dont le système d'exploitation est installé avec une image monolithique, comme les Raspberry-PI par exemple. L'API Yoctopuce utilise les numéros de série Yoctopuce pour communiquer et le numéro de série d'un VirtualHub est créé à la volée à partir du *hostname* de la machine qui l'héberge.

12.9. Commandes ignorées

Si vous avez l'impression que des commandes envoyées à un module Yoctopuce sont ignorées, typiquement lorsque vous avez écrit un programme qui sert à configurer ce modules Yoctopuce et qui envoie donc beaucoup de commandes, vérifiez que vous avez bien mis un `YAPI.FreeAPI()` à la fin du programme. Les commandes sont envoyées aux modules de manière asynchrone grâce à un processus qui tourne en arrière plan. Lorsque le programme se termine, ce processus est tué, même s'il n'a pas eu le temps de tout envoyer. En revanche `API.FreeAPI()` attend que la file d'attente des commandes à envoyer soit vide avant de libérer les ressources utilisées par l'API et rendre la main.

12.10. Impossible de contacter les sous-devices par USB

Le but du YoctoHub-Wireless-SR est de fournir une connectivité réseau aux sous-modules qui lui sont connecté, il ne se comporte pas comme un hub USB. Le port USB du YoctoHub-Wireless-SR ne sert qu'à l'alimenter et le configurer. Pour accéder aux modules connectés au hub, vous devez impérativement passer par une connexion réseau.

12.11. Network Readiness coincé à 3- LAN ready

Vérifiez que votre connexion Internet sortante fonctionne et que vous n'avez pas un callback invalide défini dans la configuration to YoctoHub-Wireless-SR.

12.12. Module endommagé

Yoctopuce s'efforce de réduire la production de déchets électroniques. Si vous avez l'impression que votre YoctoHub-Wireless-SR 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 débarrasser de votre YoctoHub-Wireless-SR, 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>

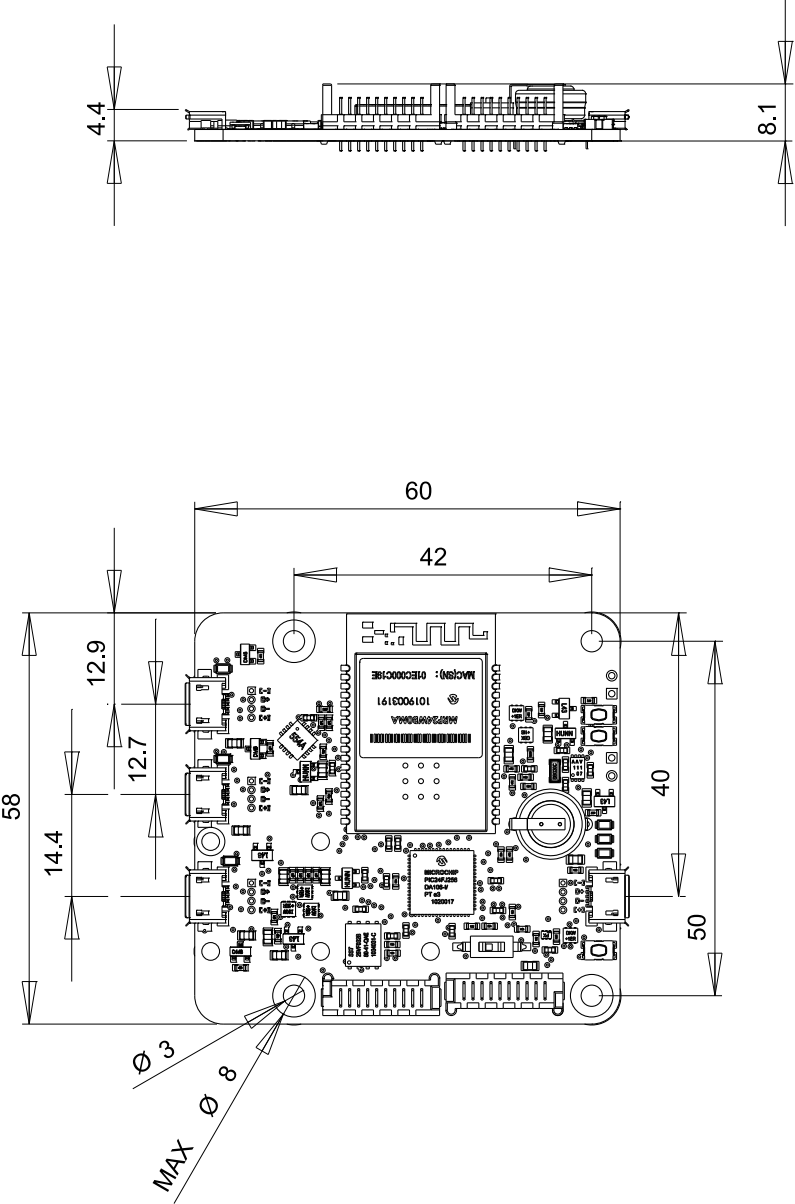
13. Caractéristiques

Vous trouverez résumées ci-dessous les principales caractéristiques techniques de votre module YoctoHub-Wireless-SR

Identifiant produit	YHUBWLN2
Révision matérielle [†]	Rev. C
Connecteur USB	micro-B
Epaisseur	8.1 mm
Largeur	58 mm
Longueur	60 mm
Poids	30 g
Canaux	3 ports
Courant Max (continu)	2 A
Classe de protection selon IEC 61140	classe III
Temp. de fonctionnement normale	5...40 °C
Temp. de fonctionnement étendue [‡]	-30...85 °C
Consommation USB	110 mA
Connection réseau	802.11b
Conformité RoHS	RoHS III (2011/65/UE+2015/863)
USB Vendor ID	0x24E0
USB Device ID	0x003A
Boîtier recommandé	YoctoBox-HubWlan-Transp
Code tarifaire harmonisé	9032.9000
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-Wireless

A4

Scale
1:1
Echelle